NODE.JS • SNAPPY UBUNTU CORE • QT + MORE!

LINUX VOICE

TAKE BACK YOUR PRIVACY

Hide from snoopers • Encrypt your email • Browse privately
Secure instant messaging • Protect SMS messages

LITTLE BROTHER

Cory Doctorow

On copyright, Creative Commons and why open is always better

EMULATION

Play old games

Rediscover all your old Windows games – and play them on Linux!

LINUX

Inside the kernel

What goes on in the project that powers your Linux machine

34+ PAGES OF TUTORIALS

GNOME BUILDER Write your own Gnome applications
OPENSUSE Peek into the world of our favourite German export
ASTRONOMY Use raw images to find new worlds and new civilisations...

DESKTOP DISTRO

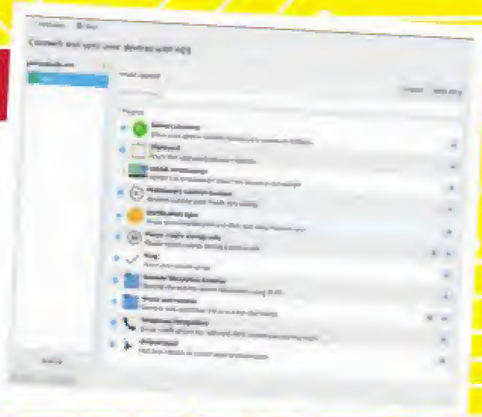
FEDORA 22

Yum is dead. Long live DNF! Plus super Gnome upgrades and more fun

CONVERGENCE!

KDE CONNECT

Link your phone to your desktop and never miss another call about PPI



FREE SOFTWARE | FREE SPEECH

ISSN 2054-3778

August 2015 £5.99 Printed in the UK

ANDREWS & ARNOLD LTD

will make you the

LINE KING

BE THE MASTER OF YOUR OWN TELEPHONE

SIP2SIM® from Andrews & Arnold allows you to treat your mobile handset as a SIP endpoint, freeing you from your desk and without the need of a smartphone app. Insert the SIM into your phone, point it to your Asterisk, FreeSwitch or other SIP server (or a commercial SIP service) and experience the reliability and call quality you're used to on a mobile, but with the flexibility of a SIP handset.

No minimum term. SIM £5+VAT and £2+VAT per month. Calls from 2p+VAT per minute.

Call 033 33 400 220, email sales@aa.net.uk or visit www.SIP2SIM.uk to find out more

HAKUNA MATATA

The ministry of truth

The August issue

THE LINUX VOICE TEAM

Editor Graham Morrison
graham@linuxvoice.com

Deputy editor Andrew Gregory
andrew@linuxvoice.com

Technical editor Ben Everard
ben@linuxvoice.com

Editor at large Mike Saunders
mike@linuxvoice.com

Games editor Michel Loubet-Jambert
michel@linuxvoice.com

Creative director Stacey Black
stacey@linuxvoice.com

Malign puppetmaster Nick Veitch
nick@linuxvoice.com

Editorial contributors:
Mark Crutch, Andrew Conway, Juliet Kemp, Michel Loubet-Jambert,Vincent Mealing, Travis Mooney, Simon Phipps, Les Pounder, Mayank Sharma, Valentine Sinitsyn.

GRAHAM MORRISON

A free software advocate and writer since the late 1990s, Graham is a lapsed KDE contributor and author of the Meeq MIDI step sequencer.

The *New York Times* has an excellent online tool that tracks its use of words, as a percentage of articles that contain them, throughout the newspaper's history (see **http://chronicle.nytlabs.com**). Use of the word 'police', for example, has remained more or less static, despite the monumental differences between the civilisations of 1850 and 2015. Search for the word 'privacy' however, and its use bumps along at around 0.05% for 100 years before ramping up in 1950 to the current high of appearing in 2% of all articles in the newspaper.

I don't think it's a coincidence that privacy usage follows that of another word, 'computer'. Privacy in 1850 could only be physically intercepted – opening a letter, or reading a diary, for example, making it a non-issue. In 2015, it's big business. From loyalty cards to browser cookies, our privacy is a commodity and we are the product. I'm not a tin-hat wearing cynic. I'm not complaining: this is a system that enables us to do some wonderful things. But I absolutely must be in control of what I want to share. And the only possible system for administering that control is open source.

Graham Morrison
Editor, Linux Voice

What's hot in LV#017

ANDREW GREGORY

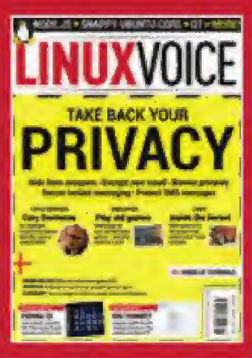
"We delve into how the kernel is created, because it's so easy to overlook the gargantuan job of putting it together" **p34**

BEN EVERARD

"Andrew Conway has written a fascinating look at using RAW photos to grab infrared data to map the stars." **p92**

MIKE SAUNDERS

"I love all things Nintendo – even geeky projects! That's why our guide to taking images with the Wiimote is my choice." **p80**



CONTENTS

August LV017

As flies to wanton boys are we to the gods; they play with us for their sport. Oh well!

SUBSCRIBE ON PAGE 64

20

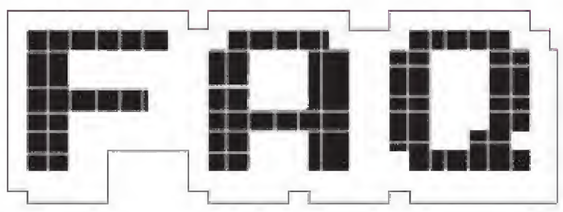
TAKE BACK YOUR PRIVACY

Many shady organisations want to know your details – keep them out, with Linux!

42

Cory Doctorow

The information age's heir to Orwell on open source, copyright and Creative Commons.



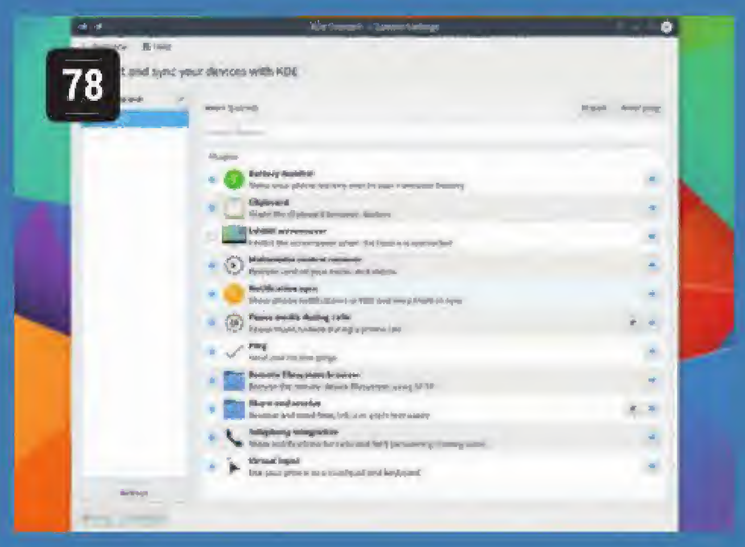
28 **PROFIT SHARE!**
Ages ago we said that we'd share our profits with the community – and here's how we did it...

30 **INSIDE OPENSUSE**
There's life in the old gecko yet – one of the oldest distros in town is still doing awesome things.

40 **FAQ: QT**
It's been part of the Free Software furniture for years, but what is Qt, and why do we need it?

REGULARS

TUTORIALS



KDE Connect: Never miss another phone call

Get desktop notifications on your phone with the magic of KDE

Build a Wiimote-triggered selfie machine

Put old hardware to frivolous use with the Raspberry Pi

Snappy Ubuntu Core: Next-gen packaging

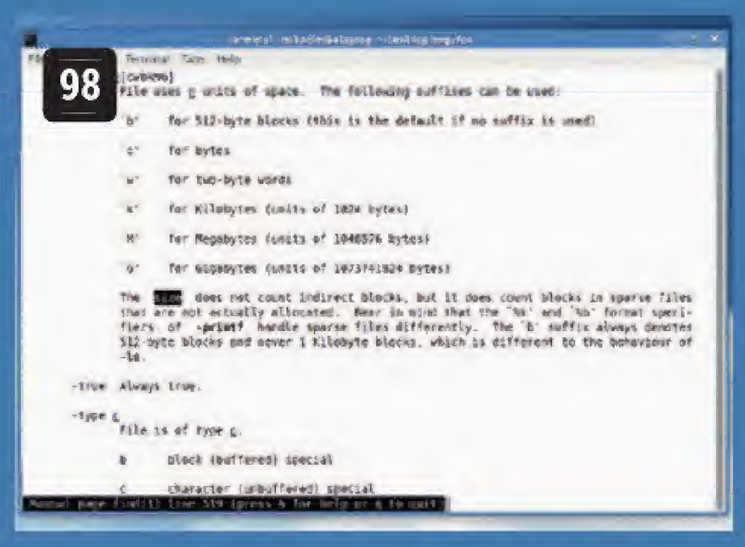
Sandbox your apps so you know they'll run perfectly every time

Run DOS and Windows games on Linux

Revisit your (or Graham's) misspent youth with old classics

Imaging in the raw with near infrared

Blow your mind with science, Python and a cheap camera



Batch jobs: Automate repetitive tasks

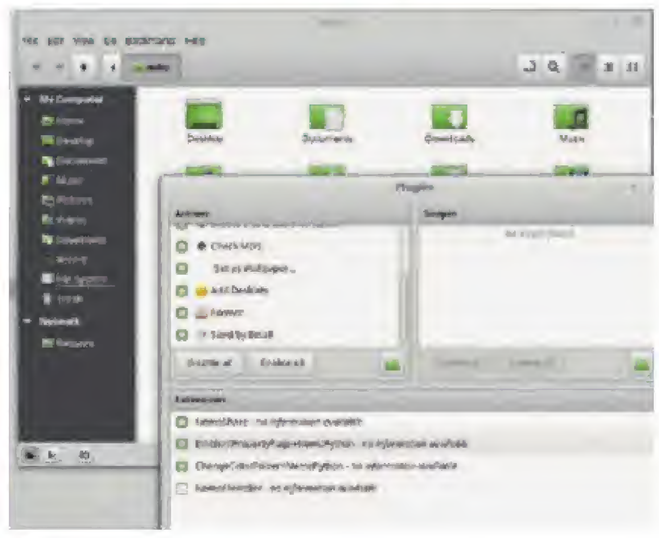
Save time and effort – make the computer take the strain

100 Olde Code: COBOL
Learn the language of business

104 Gnome Builder: Craft Gnome apps
Make applications the easy way

106 Node.js & JavaScript
Create a secure chatroom

REVIEWS



50 Cinnamon 2.6
Mint's solution to the desktop crisis is all grown up and ready to take on the big boys.

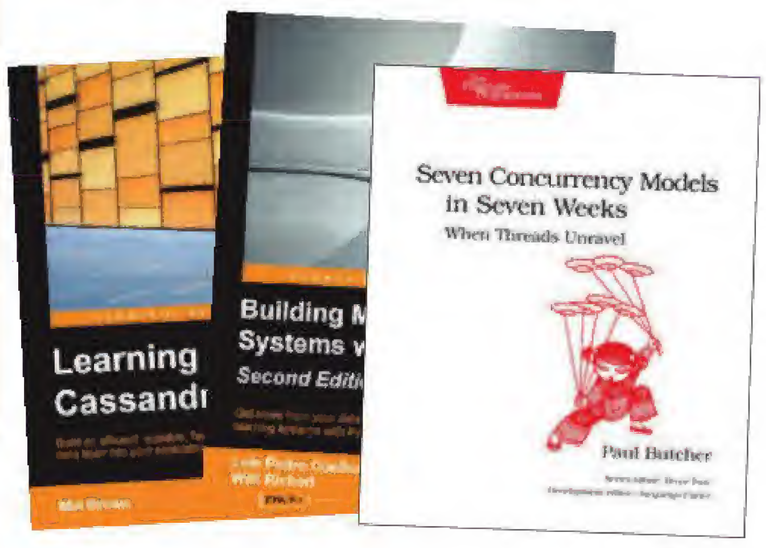
52 Fedora 22
Despite many changes behind the scenes, this smart desktop is still flying the flag for Gnome.

53 Gnu Octave 4.0
Stop press: mathematical programming language gets an excellent new GUI!

54 OnlyOffice
All the features of Google Docs and more, without all that unpleasant evil stuff.

55 Scribus 1.5
New features galore from the latest in Free Software design tools.

56 Books Ignore the lessons of Skynet and build a learning machine + more!



NEWSANALYSIS

The Linux Voice view on what's going on in the world of Free Software.

Opinion

The business value of open source

When is something free worth more than something you pay for? All the time!

Simon Phipps is president of the Open Source Initiative and a board member of the Open Rights Group and of Open Source for America.

Why do businesses – and indeed governments, educational institutions and other "enterprise" software users – need open source? That sounds like a pretty simple question on the face of it. But when I ask that question, the response is often in the vague "it's obvious" category, with people making noises about getting the software for free. That's not the real value of open source software, though. Free as in money is a benefit, but if that's the only advantage you're peddling when you promote free software, it's going to be very quickly undermined by a quick look at the total cost of ownership – things like training, maintenance etc.

When you buy proprietary software for your enterprise, you surrender control to the proprietary vendor. They are the only ones who can change the software, so you either have to give them architectural control of your enterprise software or you have to commission custom extensions or even custom systems. Giving a software company architectural control over your systems makes your lock-in to them ever deeper, while custom software just for your business is very expensive both to create and to maintain.

In turn, lock-in to the vendor means you no longer have negotiating power when it comes to paying for service. You have two options when they set a price in successive years; you can pay what they ask, or you can stop using their software. There's no third option where you keep using the software and buy service elsewhere, as you need a licence to use their proprietary systems which you'll not keep if you don't pay. Even if you buy a 'perpetual licence' things aren't a lot better, as no alternative vendor will be able to offer service without themselves having permission – in the form of a commercial relationship – with your vendor.

Proprietary disadvantage

Curiously, although buying proprietary software involves surrendering business flexibility, ceding architectural control and losing budget control, none of these things ever seem to show up as issues in procurement negotiations (probably because these are problems for another year, and too many organisations are focussed on the short term). If you could procure software that protected your business flexibility, left you or your implementation partner with architectural control and allowed you to set your own budget priorities every year, that software ought to be preferable?

Turns out you can. Open source software comes with permission granted in advance to use the software for any purpose, have full access to the source code, make whatever changes you want and both contribute those changes to a community to share in their maintenance and give the software to anyone you want. In other words, it comes with software freedom.

These days, open source software is just as capable and comprehensive as any proprietary solution, and there are plenty of vendors who will provide you with the same implementation support, ongoing service, warranty and all the other business necessities you expect from a proprietary vendor. As long as your supplier actually delivers software freedom to you, and hasn't kept if for themselves while selling you a proprietary "enterprise edition", you then have the ability to choose whether to hire in-house experts or buy service from the vendor, leaving you in control of how much you spend, when you spend and who you pay. That's how you ultimately save.

More than that, because true open source software thrives in a diverse, multi-vendor community, there's scope for a wide range of innovative approaches to using it. As a developer, the world of open source software offers rich choices every step of the way, and because permission to use them is already granted in advance, it's OK to prototype and iterate rather than having to get special permission from your vendors for each new approach you want to try.

Open source: the business case

This is the business value of open source. It re-empowers the CIO, returning control of the budget and the enterprise architecture to them. It re-energises the developer, opening up alternatives and liberating them to prototype and iterate innovative solutions for the problems they are solving. It's about regaining control, not avoiding license fees.

Given all the extra business value delivered thanks to free software and open source solutions, surely it's the proprietary software that should be cheaper because of all its disadvantages?

"As a developer, the world of open source software offers rich choices every step of the way."

Kubuntu • Raspberry Pi • Chip • Mandriva • OnlyOffice • Firefox

CATCHUP

Summarised: the biggest news stories from the last month

1 Canonical boots out Kubuntu lead developer

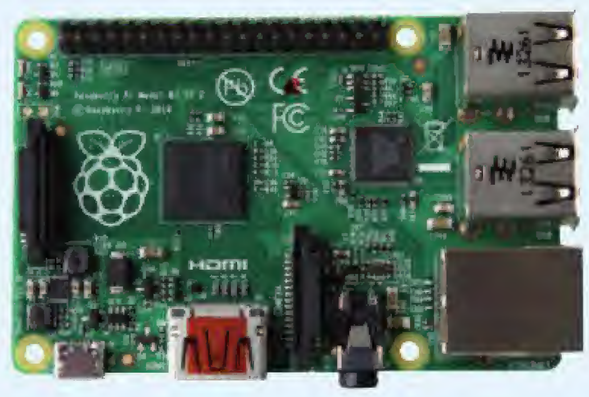
For the last two years, Kubuntu lead developer Jonathan Riddell has been asking the Ubuntu Community Council (UCC) questions about copyright and donations. Now the UCC has decided that Riddell's communication was too aggressive and confrontational, and demanded that he leave his role for a year. But the Kubuntu Council defended Riddell, and is concerned that the UCC believes it can override derivative distros. This story is still unfolding as we speak...

2 SourceForge injects ads into Windows downloads

SourceForge, once the premier host for free software projects, has taken a turn for the worse. It declared the *Gimp-for-Windows* project as "abandoned" and took it over, modifying the installer to include "easy-to-decline third-party offers", which in the real world means adware. This doesn't affect Linux users, but it could negatively impact the overall perception of free and open source software, and has caused much consternation in the community.

3 Raspberry Pi B+ price dropped down to $25

While everyone is talking about the Raspberry Pi Model 2, the older version is still going strong and is now even cheaper at just $25. But how will it fare against Chip (see below)?



4 Qemu floppy driver bug opens up security hole

PC emulator *Qemu* is often used in conjunction with hypervisors such as KVM and Xen to provide virtual machines. Now a bug has been found in the virtual floppy disk controller of *Qemu*, opening up a potentially major security vulnerability that already has its own moniker: Venom. The bug could allow code running in a virtual machine to escape the confines of the VM and affect other services running on the same machine. *Qemu* has responded quickly and released a patch.

5 Distro maker Mandriva goes into liquidation

This has been on the cards for a few years, but it's still sad news. Mandriva (formerly Mandrake), the company behind the eponymous desktop distro, has run out of cash and closed up shop. All of us at Linux Voice remember the shiny Mandrake boxed sets from the early 2000s, and the distro's excellent installer and hardware detection (back when those aspects of Linux were somewhat lacking). Still, Mandriva lives in in two forked projects: Mageia and OpenMandriva.

6 Nine-dollar Chip gadget scores big on Kickstarter

Its specs are not amazing: 1GHz CPU, 512MB RAM and 4GB of onboard storage. Plus it only has composite video output and you'll need to connect your devices via Wi-Fi or Bluetooth. But at just $9, the Chip could be the next Raspberry Pi: a tiny, cheap computer you can use in all manner of projects. Its makers were looking for $50,000 via crowdfunding, but have already snagged $1,893,355. Delivery is due at the start of next year.
http://tinyurl.com/onqqech

7 Cloud-based OnlyOffice suite goes open source

Formerly known as Teamlab Office, OnlyOffice is a suite of office applications and collaboration tools that runs on a server and presents its user interface inside web browsers. It provides competition to Google Docs, LibreOffice Online and the editing component of OwnCloud, and it's now open source. To simplify deployment, the OnlyOffice team has made the server and related tools available as Docker containers.
www.onlyoffice.org

Rank	Country	Gold	Silver	Bronze	Total
1	USA	46	29	29	104
2	China	38	27	23	88
3	Great Britain	29	17	19	65
4	Russia	24	26	32	82
5	South Korea	13	8	7	28
6	Germany	11	19	14	44
7	France	11	11	12	34
8	Italy	8	9	11	28
9	Hungary	8	4	5	17
10	Australia	7	16	12	35
11	Japan	7	14	17	38
12	Kazakhstan	7	1	5	13
13	Netherlands	6	6	8	20
14	Ukraine	6	5	9	20
15	New Zealand	6	2	5	13
16	Cuba	5	3	6	14
17	Iran	4	5	3	12
18	Jamaica	4	4	4	12
19	Czech Republic	4	3	3	10
20	North Korea	4	0	2	6

8 Firefox gets new Adobe DRM decryption module

Digital Rights Management (or as we prefer to call it, Digital Restrictions Management) aims to stop people copying music and video that they view on the internet. *Firefox* maker Mozilla is in principle against DRM, but also needs to maintain market share in a world where Netflix and other media providers are dominant. So new versions of *Firefox* can download a binary blob from Adobe to decrypt DRM content – but it's still possible to get *Firefox* without it, thankfully.

DISTROHOPPER

What's hot and happening in the world of Linux distros (and BSD!).

Porteus Kiosk 3.4

Gentoo for web kiosks.

Take the rolling-release Gentoo distribution, strip it down to a 36MB ISO image, and what do you get? A distro that doesn't do very much out-of-the-box, that's for sure. But Porteus Kiosk (**http://porteus-kiosk.org**) isn't meant to be a general-purpose flavour of Linux. It's designed to power internet kiosks – those fixed-purpose machines used solely for browsing the web, like you see in libraries, museums and airports. A kiosk distro needs to be as simple and minimal as possible, so that users can't escape from the browser and start running SSH, *GCC* or (heaven forbid) *SuperTuxKart*.

When you boot up Porteus Kiosk in live mode, running directly from RAM, a graphical wizard appears asking you to configure the network (eg Ethernet or Wi-Fi, manual or DHCP) and then choose a web browser. The distro doesn't include a browser as standard, but you can ask it to download *Firefox* or *Google Chrome*. In the next step you're asked if you want to enable "remote kiosk management", which lets you store settings in a remote server and administer multiple kiosks simultaneously. It's a very useful feature if you're rolling out a bunch of kiosks and want to keep all your settings in sync.

The Kiosk Wizard lets you lock down the browser to prevent users from messing around.

The Porteus team is trying to monetise the distro by offering automatic support updates for a price, after a three month trial period. But even without them, it's a very polished, easy to use and highly configurable kiosk distro – the best of its type we've seen.

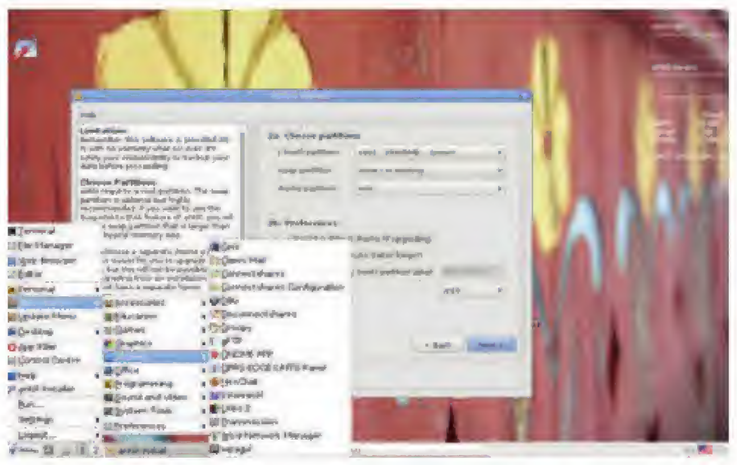
Antix 15

Lightweight and free of Systemd.

We don't mean to fan the *Systemd* flames here, and we know that many people are happily using that init system replacement in their distro. But equally, we know that many long-time Linux fans are looking for a more traditional Unix-like experience, with *Bash*-coded init scripts and plain text logs, so we always keep an eye out for *Systemd*-free distros. There aren't many around now, but a few pop up from time to time, such as Antix (**http://antix.mepis.org**).

This is a Debian Testing-based distro that's designed for general purpose desktop use, but is targeted at older machines. It's available as a 670MB ISO download – so it will burn to a CD-R – and can run on PCs with as little as 64MB of RAM. Of course, on such a low-spec machine you can't expect an amazing experience, but to revive an old box for the kids to play around with, it's ideal. If you want to do something more productive, such as web browsing or image editing, you really need at least 256MB RAM.

Given that Antix is based on Debian, which has recently switched to *Systemd*, how can the distro stick with traditional *SysVinit* for future releases? Well, the Antix team is



Antix runs the svelte *IceWM* window manager, and also includes *Dillo* for web browsing.

building on the work of the Devuan project, the Debian fork that claims to "preserve init freedom". It remains to be seen whether Devuan is a serious long-term project or just a knee-jerk reaction to *Systemd* from a bunch of frustrated Debian users, but Antix is a sign of their efforts being appreciated.

News from the *BSD camps

What's going on in the world of FreeBSD, NetBSD and OpenBSD.

While the major desktop environments of KDE, Gnome and *Xfce* tend to run fairly smoothly on most of the *BSD variants, the experience is often far from perfect. In many cases the developers simply assume that the target is a Linux system, and therefore add snippets of Linux-specific code that the *BSD teams later have to work around. It's not the end of the world, but it means that integration with hardware and system settings is often lacking when running these desktops on a BSD flavour.

To combat this, the makers of PC-BSD (a desktop-oriented FreeBSD spinoff) have been working on Lumina, an environment created from scratch (using C++ and *Qt*), which aims to be much more integrated with the operating system. It uses *Fluxbox* for its window manager and a handful of standard X utilities, but otherwise is a unique project. Examples of integration include the ability to restore ZFS snapshots directly in the file manager, and a configuration tool that uses FreeBSD's *sysctl* utility to change things like screen brightness.

When Fluxbox is replaced Lumina will offer even more cosmetic customisability, including font scaling, plus compositing support for transparency effects and improved keyboard shortcuts.

Still, Lumina aims to be portable to other BSD flavours – and indeed Linux – and therefore avoids dependencies on D-Bus, *PolicyKit*, *Systemd* and other frameworks. In addition, the desktop isn't bundled with any specific end-user applications, because the major FOSS desktop apps such as *Firefox* and *LibreOffice* run well on FreeBSD.

Currently the PC-BSD team is working on a replacement window manager for Lumina. If it's done right, this could really boost PC-BSD as a great standalone OS.

SUSE 5.2 – A boxed set full of Linux love

Over on page 30 we look at the current state of the OpenSUSE project, and see what the distro team is working on. We have a soft spot for SUSE, because, as with Red Hat, it was the first distro for many new Linux users in the late 1990s. With most of us running painfully slow dialup modem connections, the boxed sets packed with CDs and manuals were a joy – absolute treasure chests of free software and Linux goodness, delivered to your doorstep.

For some top quality retrostalgia, we dug out SUSE Linux 5.2 (from March 1998) and attempted to install it. But not on a real PC, mind you – that would probably be impossible given the changes in hardware over the last 17 years. Instead, we tried it in trusty old PC emulator *Qemu*, and for the most part it went well. It's fascinating to look back and remember how complicated distro installers were back then. You had to think about blocks on your hard drive, inodes on your filesystem, what device node to use for your mouse, and so forth. A long way from today's graphical installers where you can get a complete desktop distro running with fewer than 10 mouse clicks...

SUSE 5.2 installed correctly, but we had major problems setting up X – which was also a bane at the time. *XF86Setup* created a usable VGA16 configuration for us, and tested it, but when we tried to run X independently (via **startx**), it bailed out with "no valid modelines found" (even though we had specified one). Back in the day, using the wrong modelines could make your monitor explode, but this isn't a problem now. We'll keep battling away with it though! LV

SUSE had fantastic boxed sets back in the late 90s. Here's the German edition of version 4.2 (image: Samsara on Wikipedia).

GAMING ON LINUX

The tastiest brain candy to relax those tired neurons

GOOD OLD GAMES

Michel Loubet-Jambert is our Games Editor. He hasn't had a decent night's sleep since Steam came out on Linux.

It makes sense that many Linux gamers tend to stay away from DRM, and thus Steam as a platform. Luckily we have **GOG.com** and the countless Linux titles it offers DRM-free, both new and old. It has recently released classic games such as *Cyberia 2*, *Star Wars Rebel Assault*, *Star Wars Tie Fighter & X-Wing*, *Toonstruck* and many more thanks to FOSS projects such as *DOSBox* and *ScummVM*. The store isn't just limited to older releases though; you can find more recent games like *The Witcher 2* and *Pillars of Eternity* there completely DRM-free.

Even more exciting is that the desktop client, *GOG Galaxy*, has reached open Beta and the Linux version is soon to follow. This client adds many of the features found on platforms like Steam, such as achievements and in-game chat, but is completely optional, meaning you don't have to use the client to buy games from GOG.com. This has been eagerly anticipated by many who want to manage their games without all the desktop clutter, and to get some of those Steam-like features.

Somewhat disappointingly though, GOG.com has been extremely vague with the release date of the Linux version, when we had previously expected that it would be released alongside the Windows and OSX versions. In the meantime though, non-DRM gamers should pay them a visit, while Steam gamers should note that there are a lot of games there that aren't on Steam, so it's worth visiting.

Spec Ops: The Line

Turns the traditional modern shooter genre on its head, then makes it scream.

The problem with having a game based on *Heart of Darkness* is that if you've read the book or watched *Apocalypse Now* then you pretty much know what the main premise is going to be. On the other hand, we get a game that reinvents the traditional modern shooter genre, delivering instead something that shows the grim brutalities of war.

The main character is pretty well fleshed out, and having the third-person perspective helps to deliver a character-driven plot. He and his two US Army companions are tasked with finding a missing squadron and its leader in the Middle East, after it was presumed that a sandstorm wiped them out, along with the city of Dubai.

That all sounds pretty standard up to that point, but rather than being a shooter, *Spec Ops: The Line* is the definitive anti-shooter, ridiculing the glorification of war. Instead of rewarding the player when taking questionable actions, it goes out of its way to show you what a terrible person you are. At one point the game has you wipe away an entire army of people using white phosphorus from the comfort of a computer

Captain Martin Walker's mental condition gradually deteriorates as the horror unfolds before him.

screen, then shows you the resulting carnage.

Perhaps the best executed aspect of the game is allowing for a great deal of fun in all the shooting and tactical combat, before pulling the player back like a disapproving parent, reminding you what you are in fact enjoying. *Spec Ops* then isn't for the faint of heart, but for those who want a great story and change from the standard modern FPS games, it's certainly worth getting.

Website http://store.steampowered.com/app/50300 **Price** £19.99

The game deals with a lot of difficult issues, from clandestine CIA operations to PTSD.

"Spec Ops: The Line is the definitive anti-shooter, ridiculing the glorification of war."

Victor Vran

An atmospheric action-RPG from the developers of Tropico 5.

This game is a great example of Early Access done right, providing an already entertaining game to its players, which the developers can then use to get feedback and make improvements before release, with lots of community involvement in the meantime.

Victor Vran puts gameplay above all else; the combat mechanics are solid and incredibly satisfying, while having a wide range of weapons, customisation and character progression creates just the right balance between action and RPG.

The two main elements missing from the Early Access version at the time of writing are voice acting and cutscenes, but the developers were kind enough to include placeholder images and a few written lines to give players an idea of what the story will be like.

Not revealing much of the story yet is another good move, allowing Early Access players to enjoy themselves while not spoiling the full experience when it comes around this summer.

Website http://store.steampowered.com/app/345180 **Price** £15.99

Levels are well crafted and provide a great backdrop for all the monster slaying.

Windward

A procedurally generated trading and exploration game with pirates!

This fun little trading and exploration game is as addictive as they come. You control a single ship trading goods with different towns, but as the game goes on there's the growing threat of rival factions and even pirates to fight on the high seas.

There's a tonne of things to do and many different ways to play, from going the diplomatic route and slowly building up the ship to carry more guns and cargo, or simply take on as many pirates and factions as possible. Both options (and everything in between) are fun and rewarding; the ship-to-ship combat is great, while performing quests for towns and seeing them grow gives a nice sense of development.

Windward is magnificently simple and easy to get to grips with, while providing plenty of challenge as it goes on. For those who want an even greater challenge, the online mode against other players is probably the most fun, though the game is worth it for the offline content alone. it is highly recommended to those who enjoyed games like *Taipan!* or *Sid Meier's Pirates*.

Windward's aesthetically pleasing maps are full of adventure.

Website http://store.steampowered.com/app/326410 **Price** £10.99

ALSO RELEASED...

Vertiginous Golf

This weird and wacky mini-golf game is a lot of fun and one of those rare cases where we get a local multiplayer game outside the console space. *Vertiginous Golf* is set in a dystopian steampunk world and provides some nice features like a map editor and a rewind ability to be able to retake shots. It's great to play at home with friends or against others online. http://store.steampowered.com/app/272890

Lux Delux

This highly customisable Risk-like game is a lot of fun and provides a different experience every time through all its options. The game boasts over 900 maps and an active community of players doing modding through an open-source SDK and built-in map editor. Online and even local bot games can be very challenging, while the cross-platform multiplayer and league tables exist to ensure plenty of competitive and nailbiting experiences. http://store.steampowered.com/app/341950

Euro Truck Simulator 2: Scandinavia

Everyone's favourite simulator game just got its second major DLC, adding a bunch of Scandinavian towns to visit, and also new cargo and companies. The new scenery is very pretty and the developers have done a great job of capturing the region's natural beauty. At the same time, a new patch has come out adding, among other things, an improved day/night cycle and weather system, adding better realism to the world. http://store.steampowered.com/app/304212

LINUX VOICE YOUR LETTERS

Got something to say? An idea for a new magazine feature?
Or a great discovery? Email us: letters@linuxvoice.com

LINUX VOICE STAR LETTER

LINUX ON THE HIGH STREET – A RESULT!

Further to my letter in LV014, it appears that HP has decided to release three Ubuntu laptops in the UK via Ebuyer: **www.theinquirer.net/inquirer/news/2406977/canonical-and-ebuyer-team-up-to-bring-ubuntu-powered-laptops-to-the-uk**.

The entry-level HP ProBook 255 (£199.98) has an AMD A4 processor and 4GB RAM, the ProBook 355 has an AMD A8 and 4GB, and the top-of-the-line HP ProBook 455 has an A10 with 8GB. All have Ubuntu 15.04 installed. Apparently HP has the "XP holdouts" in its sights:

All recent CPUs support virtualisation, so all that is needed is a Linux laptop with a virtual version of XP (or Vista/Windows 7/Windows 8) running inside it, courtesy of *VirtualBox*.

Linux would take care of secure online transactions and a Windows-only program such as *Adobe Photoshop* could run offline in Windows XP.

Godfrey Green, Cardigan

Andrew says: Well played Ubuntu! We can't fault Canonical for its ambition and attitude – the home PC market is there for the taking, and we really hope that a Linux vendor or vendors step up to the challenge and target the (as you say) XP holdouts. They don't want Vista, they don't want Windows 7 – let's show them how good Linux can be.

It's bloody brilliant to see Linux (in the shape of Ubuntu on this HP ProBook 445) as a retail offering.

RSYNC EH?

Just a quick note on the rsync article from the June 2015 issue. At one stage, you introduced the directory "bodhi" in the text, in order to explain the trailing slash. I know what you meant, but it was a bit confusing. Maybe a clarification in the next issue would be good.

Stevan Akrap

Andrew says: Now that I've gone back and had a look at the section in question, I see what you mean. They code example gave an example directory as **home/mayank**, and we then referred in the text to **home/bodhi**, and explained what would happen if we added a slash to the end, as in **home/bodhi/**. Thankfully the technical point about using a trailing slash is unchanged; you're right though that the change of directly names could be confusing. I guess Mayank, who wrote those excellent words, must have reinstalled Linux with the username 'mayank' instead of his usualy username 'bodhi'. That's one of the perils of constantly tinkering...

Bodhi Linux, not to be confused with **/home/bodhi** (or **/home/bodhi/** for that matter)

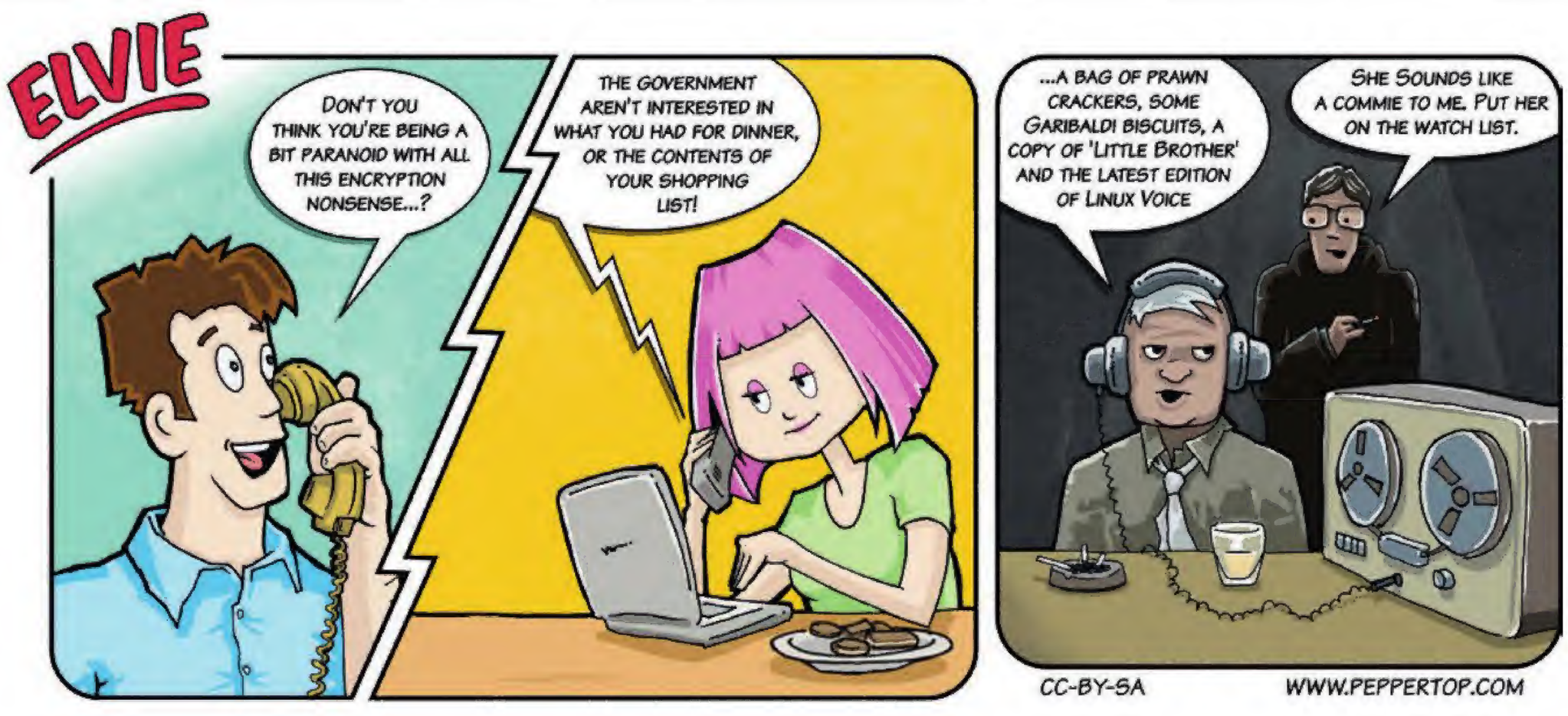
FULL CIRCLE

In issue 16 Sarah McKie asked about beginner's guide to *LibreOffice*, she could do worse than take a look at **Fullcirclemagazine.org**; this is a monthly community magazine about Ubuntu and its derivatives. They have been running a regular *Open/LibreOffice* tutorial for many years and are currently on part 48. There are also some user guides on the *LibreOffice* website but as I've not used these I don't know how user friendly they are. A look on the web should produce other resources both free and at a cost that might be useful to Sarah. As Sarah states in her letter, *LibreOffice* is a powerful application which many of us underutilise, but with patience and time many of its capabilities can be uncovered.

While I understand you're running a magazine that you sell, you still have a limit to what you can cover each month, so maybe a feature about community resources, to help readers find other sources of information they can access, might be helpful.
Tony Hughes, Blackpool

Andrew says: LibreOffice does indeed have excellent documentation, which is something we usually highlight whenever we review the most recent version. The flipside of this is that it's against the spirit of free software to build on someone else's work and effectively keep it secret by not documenting it properly. Put like that, community resources like Full Circle are doing superb work, and deserve all the recognition they get.

Full Circle is doing super work – download the latest edition for free today.



DVDS ON LINUX

Once you know which packages to install and the appropriate shell script to run, it's easy enough to watch DVDs on a Linux system.

Not sure if you've already published articles about this but these things interest me:-

- Creating DVDs with home grown content
- Being able to test the above DVDs before burning a DVD-R
- Recommendations for packages to rip DVDs and to manage DVDs
- Being able to play back ripped DVDs on systems that don't have a DVD drive

Best wishes,

Ian Bruntlett

Graham says: In the caverns of old publishing houses, you can still hear grumpy old publishers say that covering video editing in magazines doesn't sell. But I think you've made some good points, and we should do something, especially as the applications for video editing and production on Linux have suddenly become awesome. We've also got our own YouTube channel, which has been a little neglected over recent months, and this will give us the perfect excuse to dust off the lights in the studio.

We've also needed to be cautious about covering DVD ripping, as its legality was dubious in the UK, but since October 2014, transferring both your music and movie collections has become legal, so we can give it some coverage with a clear conscience. Thanks for the ideas!

If the thought of Netflix introducing advertising to your account has you scared, it's time to move to *Kodi*!

NEWS FROM OUR AUSTRALIAN COLONIES

Our (Australians') glorious leader Prime Minister Tony Abbott has been giving us all the benefit of his considered opinions again, this time on the subject of whether kids should learn coding in school. It seems that this is a silly idea and 11-year-olds should not learn transferable skills.

The irony of this is that it's his party's policy to adopt coding in the Australian school curriculum; he was asked about it by a member of the opposition, and instinctively chose to attack. I saw something similar on a UK news program, in which the host (a man in his 60s) mocked the idea of children being taught programming languages, presumably because he'd learned Latin at grammar school and hit had never done him any harm.

Luke Milosevic, Melbourne

Andrew says: The grumpy old man you remember is Jeremy Paxman, a professional grumpy old man, and

therefore easy to ignore (in fairness, the person they had on to promote programming literacy didn't seem too convincing – **https://www.youtube.com/watch?v=-7x7GYItzS4**).

Tony Abbott, on the other hand, has a responsibility to know what he's talking about or keep his mouth shut. The thing that baffles me is that, if you add a bit of Python to the curriculum and kids don't like it, what's the worst that can happen? It's only a kind of algebra – either you get it or you don't, but early exposure is vital.

"But it doesn't mean anything!" Ah Jeremy, nature is a language, can't you read?

AU REVOIR, MANDRIVA

Raise a glass, oh my brothers, to our fallen comrade: Mandriva is no more. After years in the doldrums, one of the foremost Linux vendors has gone the way of all flesh, leaving us with only memories. I suspect a lot of your readers will share with me early memories of Mandriva (or Mandrake as was) as a first desktop Linux. Before Ubuntu came along and forced everyone to take new users into consideration, Mandrake was the only Linux distro to sugar the pill of installing Linux, making it easier for converts to Linux.

Dave Moran, Basingstoke

Andrew says: The first Linux I tried was a version of Mandrake; I don't remember the version, because it didn't work. I moved on to Ubuntu 5.04 and didn't look back. But for people who tried Linux a year or two before the first few brilliant Ubuntu

releases, Mandrake did a lot to make Linux easier. And for a while it seemed at though Mandriva was going to make a success of commercialising Linux, with some big deals in the French public sector, it never really happened. Never mind though; the company may have fallen, but the code lives on. Viva Mandriva!

So farewell then, Mandriva. Thanks for all the socialist-realist penguins.

Email andrew@linuxvoice.com to advertise here

The elephant in the room is Amazon. AWS is by far the leader in the public cloud, with five times the cloud capacity of the next 14 competitors combined.

openstack SUMMIT 2015

Travis Mooney reports from from the clouds in Vancouver.

Linux has grown up a lot since 1992, from a hobbyist OS to powering datacentres, smartphones, quadcopters and basically everything else. Any technologist worth their salt knows that Linux displaced traditional Unix, and pushed a lot of BSD (and even Windows) installs aside. OpenStack has done the same thing among cloud computing platforms, taking the market by storm to become the *de facto* technology.

The crowds and marketing money going into OpenStack remind me of the explosion around Linux in the late 90s. If anything, the OpenStack crowds are more intense, the sessions more packed, and the vendor give-aways way better. And that's because OpenStack is a maturing technology, but the market isn't saturated yet, and as a result companies are making a lot of money off it.

Maturity is a double-edged sword in the technology world. The FOSS market survives largely on consulting, services, and paid development work. As software matures, it gets easier to use, gains additional features, and has greater market penetration. Making software easier to use decreases the requirement for a lot of supplementary services (consulting, especially). But as the market grows, so do the number of opportunities. This is a balance that a lot of FOSS companies know well — Canonical has been quite open about riding the OpenStack wave to significant success.

What is OpenStack, anyway?

Like Linux was as a three year-old (in 1995 I had an ancient Slackware install without X11), OpenStack is new, and can be a bit confusing. It is made up of a number of components which provide: compute service (through Nova), image service (Glance), storage (object by Swift or block by Cinder), identity service (Keystone), networking (Neutron), orchestration (Heat), telemetry (Ceilometer), database (Trove), elastic map reduce (Sahara), bare metal provisioning (Ironic), multiple tenant cloud messaging (Zaqar), shared file system service (Manila), DNSaaS (Designate), security API (Barbican), and wrap it all in a dashboard (Horizon). Just to make things more open,

and a bit more complicated, there are different drivers for various components, such as a choice of underlying filesystem driver for the storage components, or network driver for Neutron.

When you put all these programs together, you can create a cluster of machines that behave very much like any public cloud service, but completely under your control. Although you still have to procure and provision servers, you gain most of the flexibility of cloud virtualisation, can run multiple tenants on the same installation, and can have automatic scaling of systems based on demand. Adding hardware to scale out services also becomes less difficult. You can even leverage public clouds that use OpenStack to create hybrid clouds — for example to offload public web services during peak hours.

Hot topics

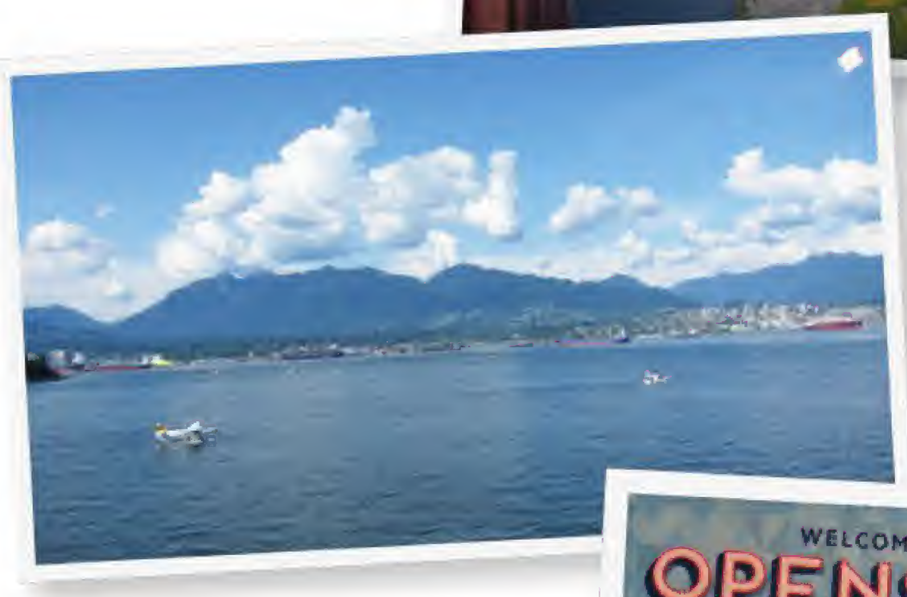
And much like the early days of Linux, there are a lot of companies building on the OpenStack ecosystem. OpenStack has buy-in from Canonical, HP, and IBM Softlayer, to name but a few. Smaller vendors at the OpenStack Summit were hawking: add-on billing engines (both for public-facing services and departmental back-billing), integration of AWS as an external public cloud, additional automation and orchestration tools, storage solutions, and many other things.

Like a lot of technology conferences, the OpenStack Summit moves, with a meeting in a new place every six months. This meeting was in Vancouver, home to Douglas Coupland, William Gibson, and Jewel Staite, so you know it has some technology credibility. It's a city of both modern glass and steel skyscrapers, and traditional brick buildings. Canada's Hollywood, many sci-fi favourites are shot in the city and surrounding countryside, including the The X-Files, BattleStar Galactica (2004-onwards), and Supernatural. And, at least during the OpenStack Summit, it's a beautiful, walkable city, filled with people so friendly that you might wonder what they're up to.

Besides the design summit, where real code was really coded, the main thrusts of the OpenStack Summit were around containers, automation and devops, and building on the OpenStack ecosystem for fun and profit.

Containers are hot on everyone's plate these days, and the OpenStack Summit featured at least three main container technologies: Docker, CoreOS, and LXD. The most established of the three, Docker, not only had overflowing talks where people competed for standing room, but featured heavily in sessions aimed at devops practitioners. CoreOS sessions were likewise overflowing, and added CoreOS technologies to Docker (or Rocket) with Kubertantes (an automation layer for containers developed at Google).

OpenStack is making a lot of money for companies like Canonical, Suse, Red Hat, HP, and IBM.



Thaks for having us, Vancouver – see you soon, Tokyo!

Shifting from fat servers to a cloud way of systems management and design was also key to the summit.

The lone LXD session was put on by Canonical, which very correctly pointed out all the ways that LXD is lighter than KVM, but that's not a surprise, as the reason that any container technology is hot is that it gives a stable platform *a la* virtualisation without as much overhead. In the end it wasn't terribly clear why Docker, Rocket or LXD would be a better choice for deployment on OpenStack, but it is definitely clear that containerisation has come to primetime.

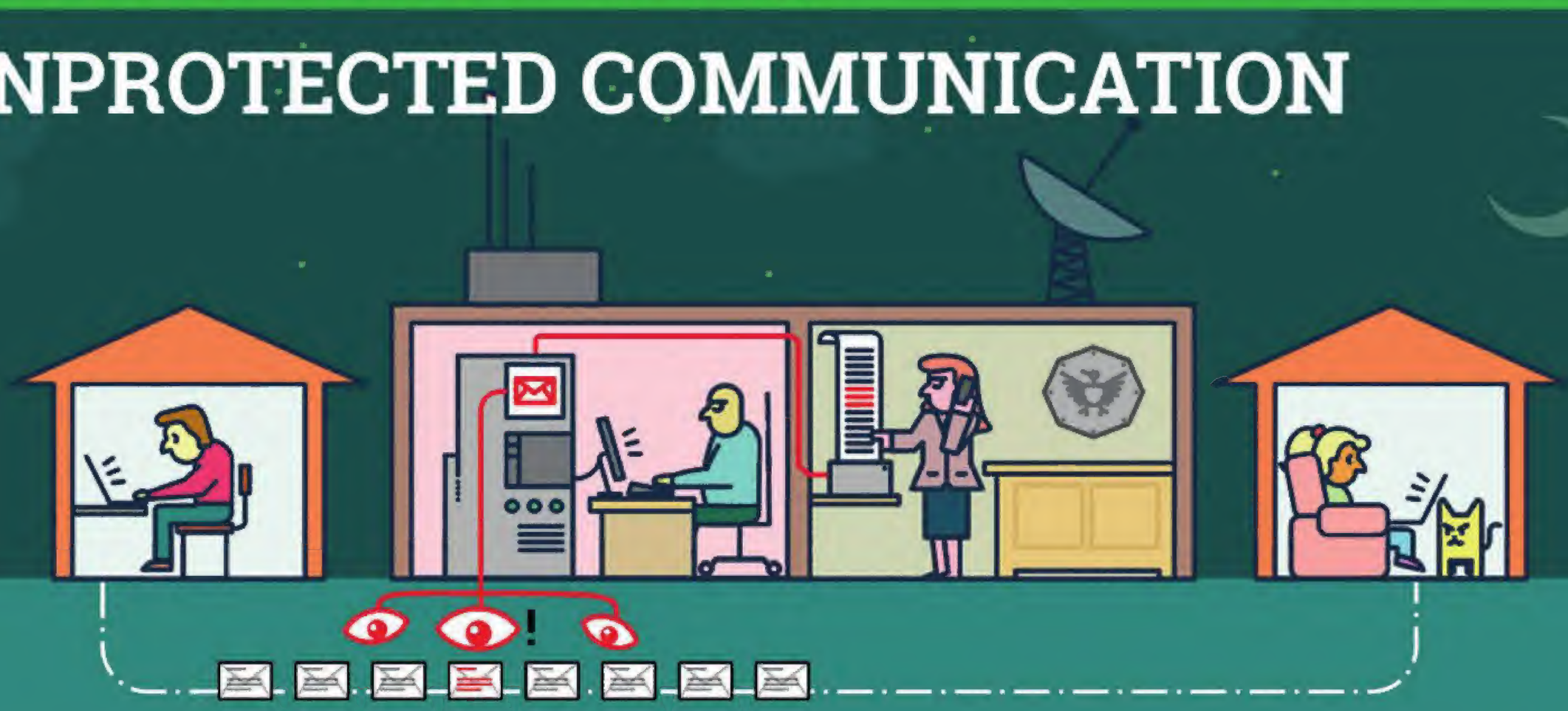
"OpenStack is one to watch; changing to a cloud mentality means that we run services, not systems."

OpenStack is definitely a hot technology to watch, and changing to a cloud mentality means realising that we run services, not systems — even if we have to make sure the underlying systems are healthy. If you want to check out the newest in private cloud technology, take a look at the current DevStack single-computer OpenStack implementation, or run a full version up on a spare cluster of six machines and take it for a test drive. It's one of the ways the future is going!

To see videos of the sessions from the OpenStack Summit in Vancouver, see **https://www.openstack.org/summit/vancouver-2015/summit-videos**.

The next OpenStack Summit will be held in Tokyo, 27–30, October, 2015.

UNPROTECTED COMMUNICATION



Mass surveillance violates our fundamental rights and is a menace to the freedom of speech! But: we can defend ourselves.

The password that protects your email is not sufficient to protect your mails against the mass surveillance technologies used by secret services.
Each email sent over the internet passes through many computer systems on the way to its destination. Secret services and surveillance agencies take advantage of this to read millions and millions of emails each day.
Even if you think you have nothing to hide: Everyone else that you communicate with via unprotected emails is being exposed as well.
Take back your privacy by using GnuPG! It encrypts your emails before they are sent, so only the recipients of your choice can read them.
GnuPG is platform independent. That means it works with every email address and runs on pretty much any computer or recent mobile phone. GnuPG is free and available at no charge.

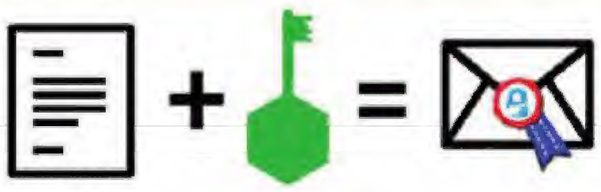
About GnuPG

Thousands of people already use GnuPG, for professional and private use. Come and join us! Each person makes our community stronger and proves that we are ready to fight back.
Whenever an email that is encrypted with GnuPG is intercepted or ends up in the wrong hands, it is useless: Without the appropriate private key it cannot be read by anyone. But, for the intended recipient - and only for her - it opens like a totally normal email.
Sender and recipient are both safer now. Even if some of your emails contain no private information, consistent use of encryption protects us all from unjustified mass surveillance.

What makes GnuPG secure?

GnuPG is Free Software and uses Open Standards. That is essential to be sure that software can really protect us from surveillance. Because in proprietary software and formats, things might happen beyond your control.
If no one is allowed to see the source code of a program, nobody can be sure that it does not contain undesirable spy programs - so-called "backdoors". If software does not reveal how it works, we can merely trust it blindly.

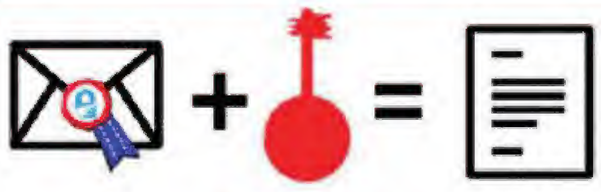
public key



encrypt

When someone would like to send you an encrypted email, they need to use your "public key". So, the more you spread/distribute your public key, the better.
Don't worry: Your public key can only be used to encrypt emails to you, not to decrypt them.

private key



decrypt

Your "private key" is like the key to the front door of your house; you keep it safe (and private) on your personal computer. Take care that you are the only one who can access it!
You use GnuPG and your private key to decrypt and read all encrypted emails that have been sent to you.

GPG-ENCRYPTED COMMUNICATION

In contrast, a fundamental condition of Free Software is to publish its source code: Free Software allows and supports independent checking and public review of the applied source code by everyone. Given this transparency, backdoors can be detected and removed.
Most Free Software lies in the hands of a community that works together to build secure software for everyone. If you want to protect yourself from surveillance you can only trust Free Software.

Practical advice

The technology behind GnuPG provides first-class protection. To ensure that your encrypted communication is not compromised for other reasons, use a strong passphrase and backup your private key. Encrypt as much as you can! By doing so, you prevent others from realising when and with whom you exchange sensitive information. Thus, the more often you encrypt your messages, the less suspicious encrypted messages will be. Be aware that the subject is transmitted unencrypted!

You can find a simple tutorial for email self-defense with GnuPG encryption here: EmailSelfDefense.FSF.org

Watch out for so-called "Cryptoparties" in your area! These are events where you can meet people that would be happy to help you in setting up and using GnuPG as well as other encryption tools at no charge.

About the fsfe

This article was created by the Free Software Foundation Europe (FSFE), a non-profit organisation dedicated to empower people in Europe in their use of technology by promoting software freedom.

Access to software determines how we can take part in our society. Therefore, FSFE strives for fair access and participation for everyone in the information age by fighting for digital freedom. Nobody should ever be forced to use software that does not grant the freedoms to use, study, share and improve the software. We need the right to shape technology to fit our needs.

The work of FSFE is backed by a community of people committed to these goals. If you would like to join us and/or help us to reach our goals, there are many ways to contribute. No matter what your background is. You can learn more about this under: **fsfe.org/contribute**

Donations are critical for us to continue our work and to guarantee our independence. You can support our work best by becoming a sustaining member of the FSFE, a "Fellow". By doing so, you directly help us to continue the fight for Free Software wherever needed: **fsfe.org/join**

What is Free Software?

Free Software can be used by everyone for any purpose. That includes free copying, reading the source code and the possibility to improve or adapt it to your own needs (the so-called "four freedoms").

Even if you "only want to use" the program, you still benefit from these freedoms. Because they guarantee that Free Software remains in the hands of our society and that its further development is not controlled by the interests of private companies or governments.

Find out more about this and how Free Software can lead us into a Free Society: **fsfe.org/freesoftware**

If you like to help spreading the word, you can order this and other leaflets under: **l.fsfe.org/promo**

fellowship of fsfe

TAKE BACK YOUR PRIVACY

Hide from snoopers • Encrypt your email • Secure instant messsaging • Browse privately • Protect text messages

The internet is a dream for snoopers. Almost every action of our daily lives flow through it at some point or another: emailed appointments, streamed TV shows, web purchases, photos shared with friends. Much of this is completely unencrypted and can be read by any of the various companies who own the tubes the data flows through. Even the bits that are encrypted are usually only encrypted between the end user and the company running the website. Once the data is uploaded, it's often mined, and the data is then sold off to the highest bidder.

Our privacy is so valuable that many companies build their business models on invading it. Facebook doesn't provide free access to a social network because it's a charity; it does it to learn about your life so that it can sell advertising space more effectively. Google doesn't index the web just to make life easier for you; it does it so it can learn about your life and sell advertising space more effectively. Twitter doesn't... well, you get the picture. The modern internet was eloquently summed up by Andrew Lewis when users complained about changes to the Digg network: "If you're not paying for it, you're not the customer: you're the product being sold".

"If you're not paying for it, you're not the customer: you're the one being sold."

Governments are also keen to investigate the finest details of our lives. They claim this is for national security and to prevent crime, but there's very little evidence that internet surveillance has ever prevented terrorism or made an impact on crime. Instead, surveillance is used to harass critics and entrench government control.

We can fight back!

Many of the underlying technologies of the internet come from a time when only a few people connected to the network, and no sensitive information got shared. If encryption and security were considered at all, they were considered a waste of resources. This can make it seem sometimes as if privacy on the internet is an impossible achievement.

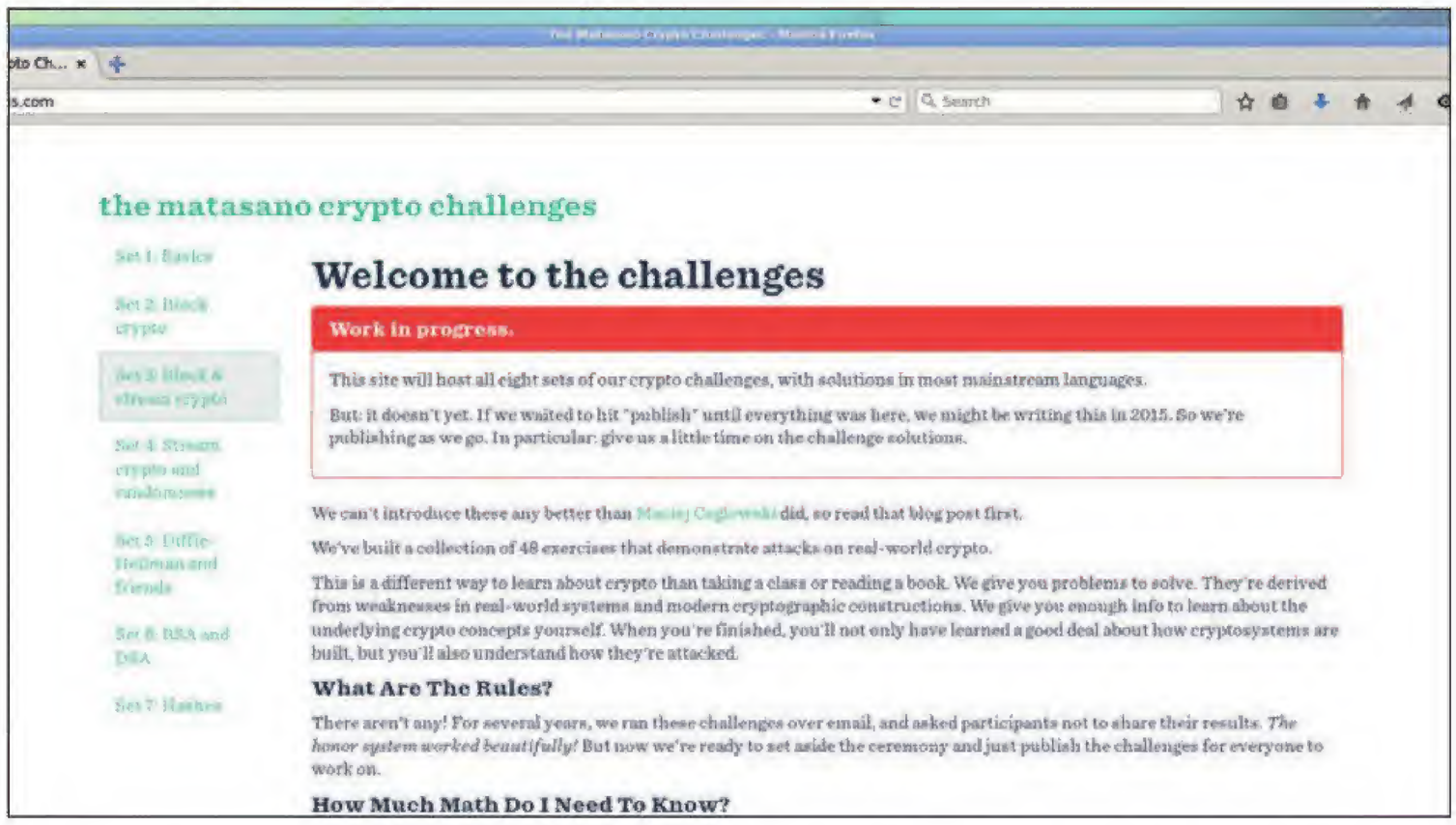
All is not lost. You can't get back the information that has already leaked out, but you can stop the invasions of privacy from continuing. All the evidence we have says that when it's used properly, modern encryption can't be broken by anyone. We'll show you how to use it properly, and help you understand what sort of security each form of encryption provides and which protocols can be trusted to keep your data private.

Encryption

A digital toolkit for keeping data safe.

Our best tool against spying is encryption. This is a complex mathematical process of changing data so that someone spying on us can't understand the data. There are three types of encryption:

- **Shared key (aka symmetric key and private key) encryption** This is where the same key is used to encrypt and decrypt information. This means that if you're communicating with someone, both parties need to know the key. This can cause a chicken-and-egg problem because you can't communicate securely until you both know the key, but you can't share the key until you have secure communications.
- **Public Key Encryption** Here, different keys are used to encrypt and decrypt data. The two keys are usually referred to as the Public Key and the Private Key. The public key is known to everyone, while the private key is known only to one person. If you want to send someone a message, you can encrypt it with their public key. Alternatively, if someone wants to digitally sign a message, they can encrypt it with their private key. Anyone can then decrypt it with their public key, and be sure that it came from the real sender.
- **Hashing (aka One-way encryption)** This is unlike the other forms of encryption because once data's been hashed, there's no way to un-hash it. The one redeeming property of hashing is that it's consistent. That means that when you hash the same value, it will always return the same result. For example, passwords should be stored hashed. When a user enters their password, the computer hashes what they enter and checks that hash against the stored value. If an attacker steals the stored password hashes (provided the password's can't be guessed), they can't actually use them.



The Matasanto crypto challenge is a great way to learn the challenges involved in encryption.

These three types of encryption are combined in various ways to form encrypted protocols that we can use to secure our data and communications.

When we talk of privacy, there are a number of different things that we could mean. It's important to understand the different guarantees that each protocol attempts to establish so we know exactly how private our communication is.

- **Secrecy**, where no-one can see the contents of our communication. However, it is possible that someone eavesdropping on a secret communication could find some information out, like who is communicating with whom. They should not, however, be able to see the data that's being transmitted between two parties.
- **Metadata secrecy**, where no-one can see who we're communicating with. They may see that a stream of information comes out of our machine, but can't track where it's going, or even what form of communication it is.
- **Non-repudiation/Tamper-proof**, a way of guaranteeing that the person who said something really said it. This is useful because it stops people impersonating other people, and
- **Anonymity.** In a truly anonymous system, no-one can tell who another person is unless they deliberately reveal themselves. In some cases this is a good thing, because it allows whistleblowers to report on issues and even the person they're blowing the whistle to can't tell who they are (and therefore can't betray them). An anonymous system could include some form of online identity system, but not a way to link that identity to a real person.

Glossary of spying terms

- **Five-eyes** An information sharing network made up of USA, UK, Canada, Australia and New Zealand.
- **Man-in-the-middle** A form of attack where the attacker positions themselves between the two parties. Here they can both sniff and alter data travelling in either direction.
- **NSA** The National Security Agency. The USA's spy agency tasked with foreign espionage and securing communications infrastructure.
- **GCHQ** Government Communications Headquarters. Britain's communications spying agency headquartered in Cheltenham.
- **Metadata** Data about data. In an email, the contents would be considered data, while the sender, recipient, subject, date and associated IP addresses would all be considered metadata.
- **FISA** Foreign Intelligence Surveillance Act. A US federal law that is used to legitimise much of the NSA's spying through a very flexible interpretation of the word Foreign.
- **Cookie** A piece of data stored by your browser that can be set by a web server. This can be used for tracking a user's session (such as keeping them logged in to a site), or tracking their movements through the web.
- **Europe vs Facebook** A legal case that's being brought against Facebook for allegedly breaching European data protection law.
- **Fingerprinting** A method of identifying a user based on the settings in their web browser – see Panopticlick by the EFF (**https://panopticlick.eff.org/index.php**).
- **Snooper's Charter** A proposed law in the UK that would bring in sweeping new powers to allow the government almost unfettered access to internet data in the UK.
- **Human Rights Act 1998** A piece of UK legislation that the current government wishes to repeal. It includes Article 8 (Everyone has the right to respect for his private and family life, his home and his correspondence).

Spying programs

Governments are vacuuming up huge troves of data on civilians...

Prism

Almost all the communication gathered by Prism is sent encrypted, but can still be gathered by the NSA because it's not encrypted for its entire journey. Take, for example, an online chat in Facebook. The messages are sent via HTTPS communication from your browser to Facebook. They're then sent from Facebook to the recipient via encrypted HTTPS, which again can't be sniffed. This means that at no point is the message transmitted unencrypted. However, Facebook has access to the unencrypted message, so it can relay it to a third party. In the case of Prism, the third party is the NSA, but Facebook also uses this information to tailor adverts. The fact that your messages are stored in Facebook's data centre also means that your messages could be read by any hacker who manages to get access to this.

The only method of defeating this form of spying is end-to-end encryption. This is where a message is encrypted by the sender and not decrypted again until it reaches the recipient.

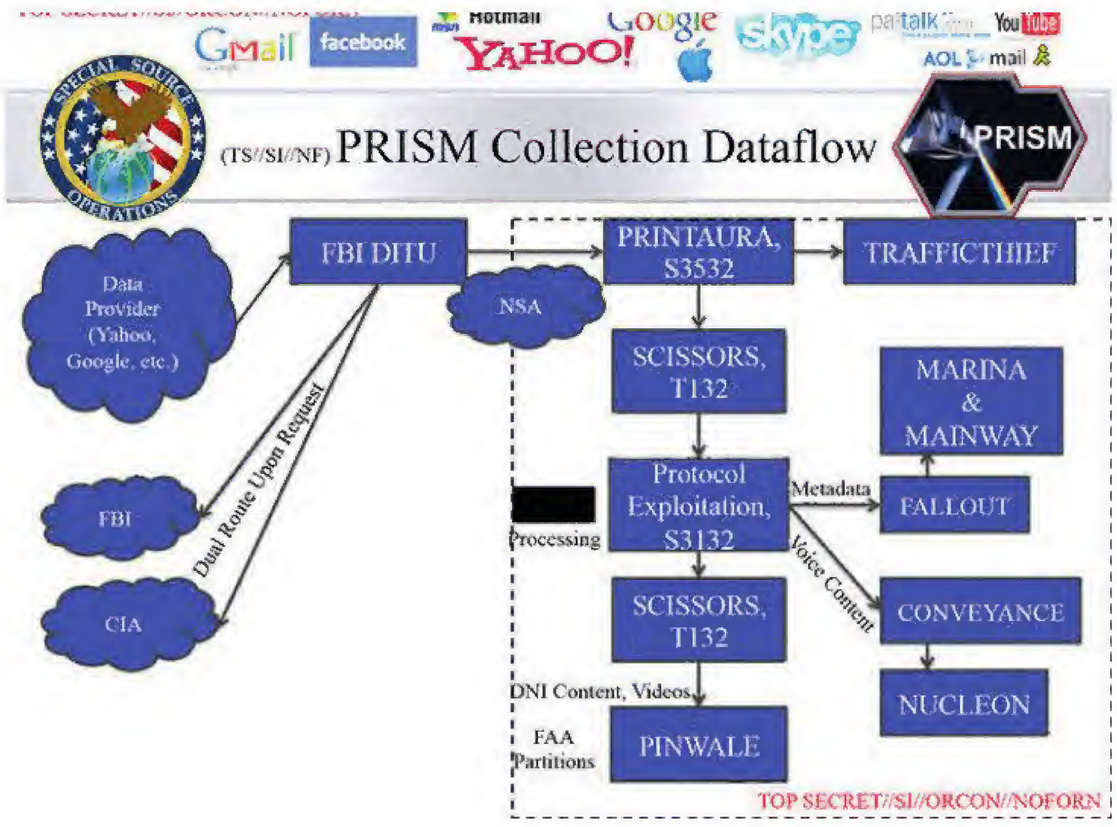
XKeyscore

XKeyscore isn't a standalone surveillance program in itself, but a front-end for all the data amassed by the NSA. It's the system that brings everything together and enables an analyst to instantly access all the information stored about another person. Everything from mundane Facebook chats to the details of your browsing history to phone calls can be accessed from a single place.

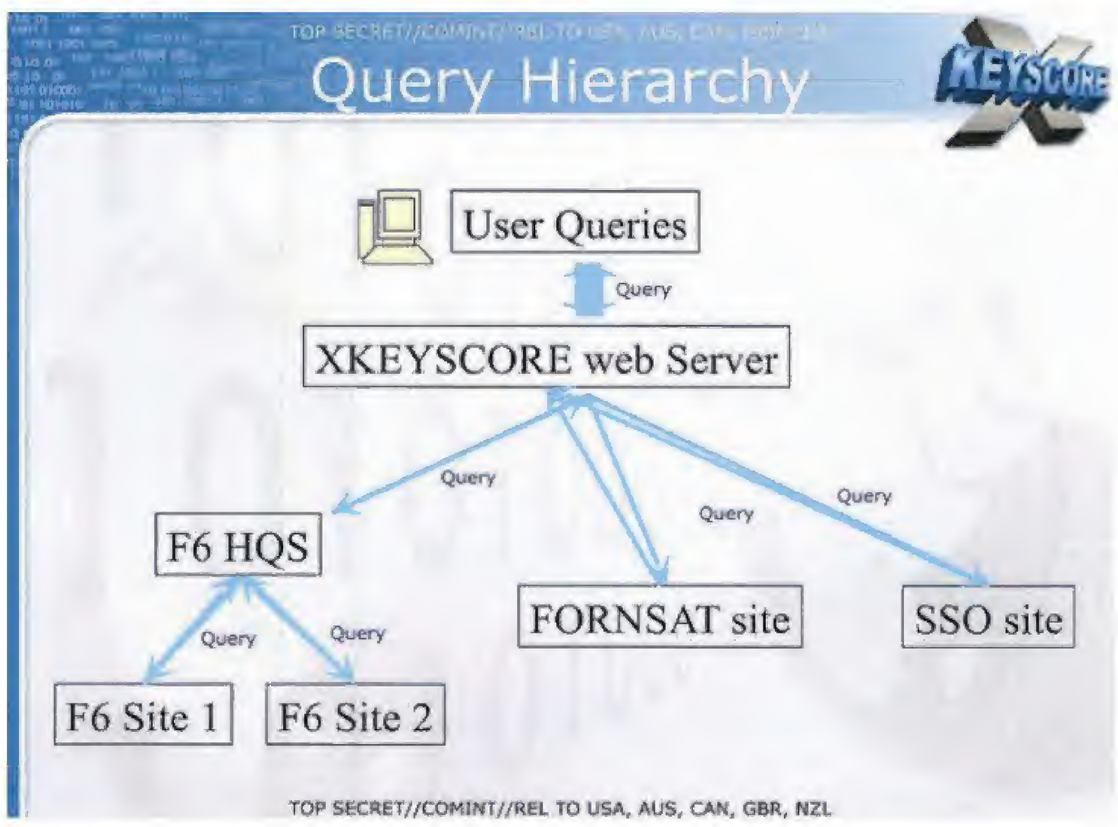
So, if you think that text-messaging a server password won't be linked to your online accounts where the server details are stored, it's time to think again. All your communications are linked to all your others (unless you're using carrier pigeons or smoke signals).

Eye in the sky

The best defences against XKeyscore are end-to-end encryption to stop a communication appearing on one of the back-end databases linked to the program, and true anonymity can mean that a particular communication can't be linked back to you.



Prism is a data source that feeds into many of the NSA's analysis tools.



Systems like XKeyscore allow agencies to analyse vast volumes of data.

Tempora

Much of the data travelling to and from the west of Europe goes via the Cornish seaside town of Bude. Here, and on nearby beaches, cables that travel to Canada, the east coast of the USA, the west coast of Africa and beyond slide below the waves and into the murky depth below. If you want to be able to sniff global internet traffic, you need a presence at Bude. It should come as no surprise that GCHQ runs one of its regional sites here, and has taps on every major cable coming into and out of Bude. These cables contain telephone (voice and SMS) data as well as internet communications.

Cream first or jam?

Project Tempora is run by GCHQ (with assistance from the NSA), and collects data directly from internet cables such as those tapped at GCHQ Bude. GCHQ has tapped at least 200 10-Gigabit cables and can process information from up to 46 of them at a time. So much data is collected through Tempora that GCHQ can't store it for long. It holds on to the full data for three days, and the metadata for 30 days. At least, that was the capability of Tempora in 2012, according to information provided by NSA whistleblower Edward Snowden.

If you're using the internet in the UK, it's almost certain that your connection will go through a GCHQ-monitored cable. If you're in mainland Europe, it's still quite likely that it will be picked up by British spooks. Many major internet companies have their European headquarters in Ireland, so most communications in or out of these data centres go through GCHQ-monitored cables as well. Anything that isn't encrypted will be extracted. Anything that is encrypted will have any available metadata extracted. There's little oversight of GCHQ, so it's impossible to know exactly what they're doing with all this data.

Dishfire

The NSA is attempting to collect every SMS message in the world using a system known as Dishfire. According to one GCHQ document, "[Dishfire] collects pretty much everything it can, so you can see SMS from a selector which is not targeted." In this context, a 'selector' is a person, so the document is showing the system collecting text messages from people who the agency have no reason to be suspicious of.

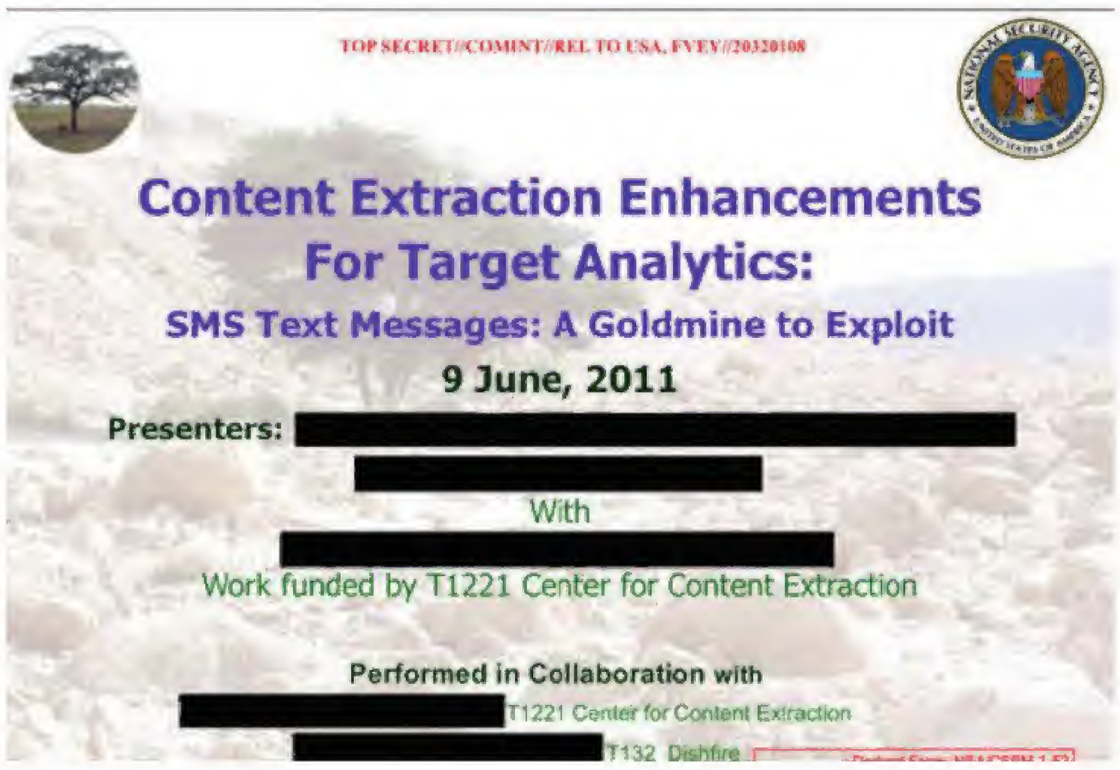
Usually, GCHQ isn't allowed to perform this sort of indiscriminate collection and analysis of British citizen's data (although oversight is minimal). However, in this case, GCHQ bypasses the Regulation of Investigatory Powers Act (RIPA), since it's technically the NSA that collects the data (it then shares it with the UK), and RIPA doesn't cover data that's shared by a foreign intelligence agency. The same loophole works the other way, since the NSA isn't allowed unfettered access to US citizens' data. Each agency collects data on the other country's citizens, and they exchange it. Thus each government follows the letter, but not the spirit, of the law.

They know where you are!

Perhaps the most disturbing aspect of Dishfire is that it doesn't just include the content of the text message, it also attempts to locate the position from which they're sent. This makes it also a database that can be used to track people (again, this is everyone, not just those suspected of wrongdoing).

We don't know how long they store the data for, but leaked slides have shown just how much information the system is gobbling up. Every day it collects:

- 200,000,000 text messages.
- 76,000 geolocated text messages.
- 800,000 financial transactions (from text–text payments or credit cards linked to phones).
- 1,600,000 pieces of information on border crossings (from roaming information texts).
- 5,000,000 missed call alerts.



Think before you text. SMSs provide a 'goldmine' to spy agencies.

Marina Mainway

In theory, US agencies aren't allowed to spy on US citizens unless they're suspected of some crime. However, there are many loopholes that the NSA exploits. Marina bypasses this restriction by not storing the content of the communication, but keeping the metadata instead.

Lawyers may argue about the difference between data and metadata, but in reality the NSA can build up a huge amount of information on someone using metadata alone.

Marina is a database of internet metadata, while Mainway stores phone metadata. Between the two, the NSA can build up a picture of your life, from your friends, to the places you frequent and the websites you visit. All this bypasses spying laws because, technically, it's not data. The difference, though, only matters to lawyers.

Boundless Informant, shown here, is a front end for Marina and Mainway.

EU Data Retention Directive

On 15 March 2006, the European Parliament and Council issued a directive stating that all member states must require telecommunication providers (such as phone companies and ISPs) to store users' data for at least six months and at most two years. This data should include things like IP addresses, email addresses, phone numbers called, text messages sent, etc.

On 8 April 2014, the European Court of Justice declared the Data Retention Directive to be invalid, though, many EU member states still require telecommunications companies to collect information about all their customers. Indeed, the UK plans to bring in even more laws regarding surveillance.

The invalidating of the Data Retention Directive by the ECJ does open up the possibility that these national laws could also be invalidated at the European level (as yet, no nation's blanket international surveillance has been tested in court).

However, a legal study financed by The Greens and the European Free Alliance concluded that, "The Court clearly rejects the blanket data retention of unsuspicious persons as well as an indefinite or even lengthy retention period of data retained." This study isn't legally binding – it's the opinion of legal experts. It states that citizens of a nation could challenge the national laws through the European Court of Human Rights. Much of the legal position on this is based on Article 8 of the European Convention on Human Rights (ECHR).

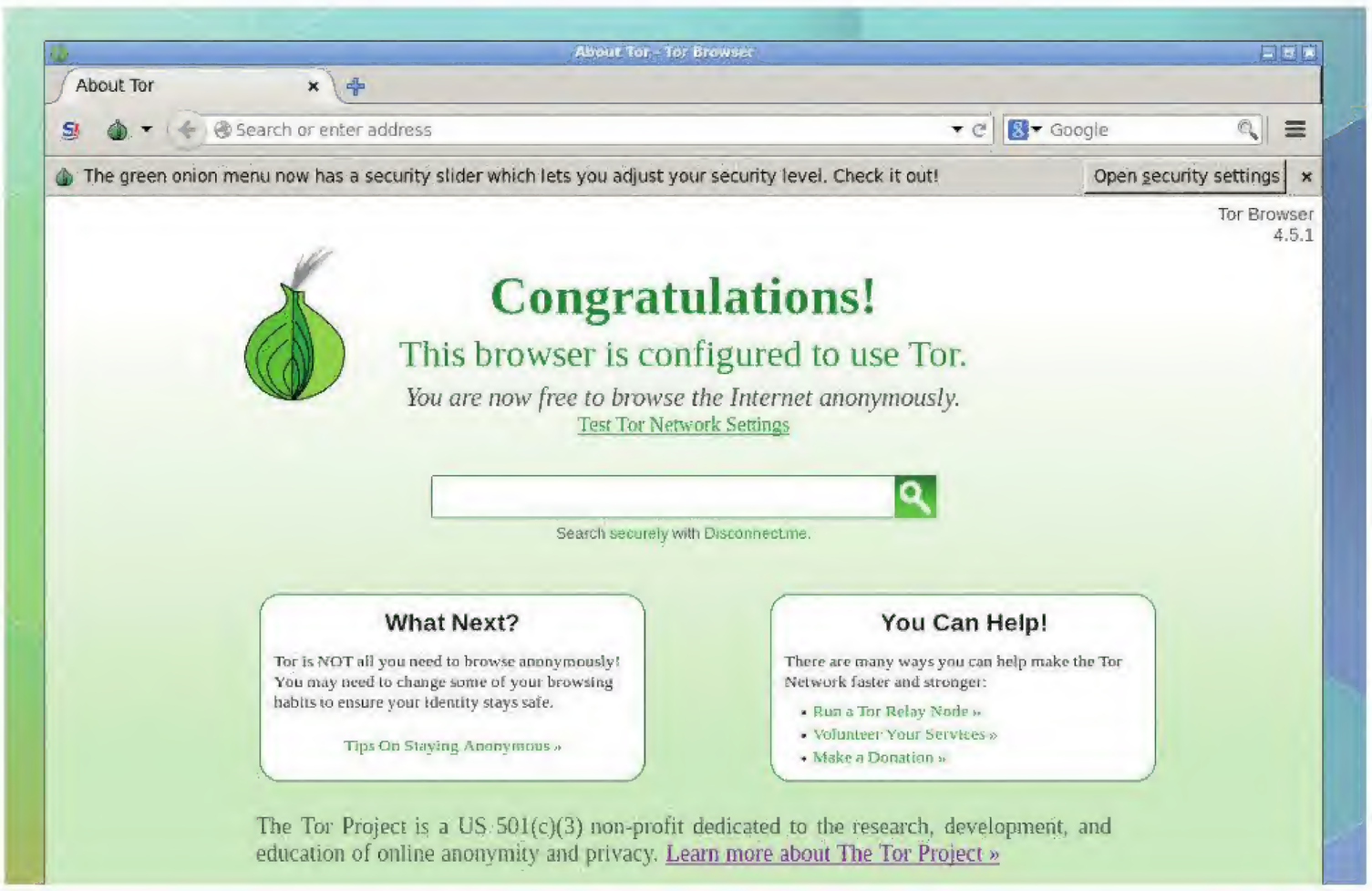
Private web browsing

Don't let everyone know what you do on the web.

Normally, when browsing the web, nothing is encrypted. All traffic is sent in the open and can be intercepted by a huge number of people. This includes the packets sent from your computer requesting data from the server, and the data the server sends back. This open communication is known as the hyper text transfer protocol (HTTP).

Even very early in the development of the web, it was apparent that not all traffic should be sent in the open. In 1995, Netscape released SSL (a layer of encryption that can be used to protect protocols that are normally unencrypted), and for the first time, browsers and web servers could communicate privately. HTTPS (the S stands for secure) is the protocol for this data exchange.

When it's working properly, HTTPS guarantees two things: no one can read the traffic, and no one can alter the traffic. There are caveats to both of these, but HTTPS is a huge improvement in security over HTTP. Anyone intercepting traffic can see what web servers you're getting data from, but not the data they send.



The *Tor Browser* allows you to surf the web anonymously, but if you don't get this screen when it first starts, then something's gone wrong and you may not be anonymous.

Rerouting connections

Web proxies are servers that you route your connection through. This means that instead of your browser sending a message to a server saying what page you want to view, it sends a message to the proxy saying what page you want to view. The proxy then requests this page from the server, and sends the resulting page back to you. If the connection between your computer and the proxy is encrypted, no-one can see what server you're requesting pages from (except the proxy itself). If the page also uses HTTPS then no-one (except the proxy) can see or alter the data from the server. However, the proxy is in an extremely privileged position. They can see just about everything you're doing on the web. Many companies that provide proxies promise that they delete logs (or don't keep them at all), but there's no way of confirming this. In many cases, proxy providers will be bound by national laws to turn over information to the authorities, or data could be stolen by hackers. In other words, proxies only provide security if the organisation running the proxy behaves well. If they don't, then proxies can provide less security than plain HTTPS.

The onion router

If you need anonymity online, the most robust option is to use *Tor*. This is a network where you communicate through a chain of three proxies. You first establish a connection from your machine to one proxy. Then, through this proxy, establish a link to a second, then through this establish a link to a third, then through the third, connect to the web. In this chain, the first proxy can see your IP address, and it can see the IP address of the second proxy you're using. The second proxy can see the IP addresses of the first proxy and the third proxy, and the final proxy can see the IP address of the second proxy and it can see the web traffic. This means that even if one of the proxies in the chain is spying on you, it can't work out who you are and what you're viewing. Of course, if an adversary controls a large portion of the nodes in the network, then they may be able to de-anonymise the traffic.

Cookies, trackers, web beacons - Following your browser

Advertising companies don't need to resort to monitoring data flowing through wires in order to track users: your web browser will tell them everything they need to know. Cookies are bits of data that can be set by a remote website and are stored on your browser. They're most commonly used to set an ID so that a website can tell which requests come from a single browser. Every time your browser requests a page from a server, it will send details of any cookies set by that domain along with the request.

When used responsibly, they're good for web users. For example, they enable web shops to follow the user as they browse the store and add items to their shopping cart. The real problem with cookies comes when a website loads content from more than one source. For example, if you go to a website with a Google advert or a Facebook-like button, your browser has sent a request to Google or Facebook, and the tracking cookies will be sent along with that request. Since a huge number of pages include content from advertising companies, these companies get a very complete picture of your browsing habits.

Most web browsers enable you to set how your browser sends cookie information at three levels: all cookies; no third-party cookies; and no cookies. The 'All cookies' option allows any advertisers to track you. 'No third party cookies' only allows cookies associated with the domain that the main web page you're viewing is from. This is a good option if you're concerned about being tracked by advertisers, but willing to accept less than 100% privacy for the convenience of websites being able to remember some information about you. Picking the 'No cookies' option may cause issues with some websites, but will give you more confidence that you're not being tracked.

The *Tor* network provides anonymity, but not security. That means if you're browsing the web over unencrypted HTTP, people will still be able to see what you're reading (or sending), but they won't know who is reading or sending it. Therefore, it's important to use additional encryption appropriate to the type of communication you want to do (eg HTTPS, GPG or ORT – see next page) in order to get both anonymity and privacy.

There are two ways of using *Tor* to browse the web: install the software on your system, or use a live distro that comes with it preinstalled. To install the software, go to **https://www.torproject.org** and select the Download link. It should automatically detect the operating system that you're running, but you'll need to make sure you use the correct link to get the 32- or 64-bit version. You should then download the **tar.xz** file, which can be installed with your normal decompression utility or from the command line with:

```
tar xvf tor-browser-linux64-4.5.1_en-US.tar.xz
```

You may find that you need to install the appropriate utilities to extract the **xz** file. You should find these in your file manager. In Debian- and Ubuntu-based systems, you'll need the **xz-utils** package. This will extract a folder called **tor-browser_en-US** (depending on your language). In this folder you'll find a file called **start-tor-browser**. This is an executable script. Depending on your file manager's setup, you may be able to click on it to run it, or you may have to use the command line. On our machine, we can run it with:

```
~/Downloads/tor-browser_en-US/start-tor-browser
```

If you don't want to install the software on your machine, or don't trust the operating system not to spy on you, then running a live distro is the best option. There are a few options, but by far the most trusted is Tails (**https://tails.boum.org**). You can run this from a CD, USB stick, or as a virtual machine. It has everything set up and ready to run, but you do need to make sure that you download any updated versions as they come out to ensure that you always have protection.

Whichever option you choose, once you've started the *Tor Browser*, you'll see that it's a modified version of *Firefox ESR* (Extended Support Release). If everything has gone correctly, you should see a green page that states, "Congratulations! This browser is configured to use *Tor*." If you see this, you can start browsing the web anonymously. However, it is worth reading the page linked as Tips On Staying Anonymous (**https://www.torproject.org/download/download.html.en#warning**) to make sure you fully understand what *Tor* does and doesn't do.

There are a number of privacy/convenience tradeoffs when it comes to web browsing, such as which cookies to accept (see boxout). It can be hard for non-technical people to understand what the issues are, and decide where to draw the line. The *Tor Browser* has a slider to enable you to increase or decrease privacy levels (and consequently decrease or increase the functionality of the browser). If you go to the onion drop-down menu in the top-left corner, and select Privacy and Security Settings, you'll get a pop-up box that lets you adjust the features you want.

"There are a number of privacy/convenience tradeoffs when it comes to web browsing."

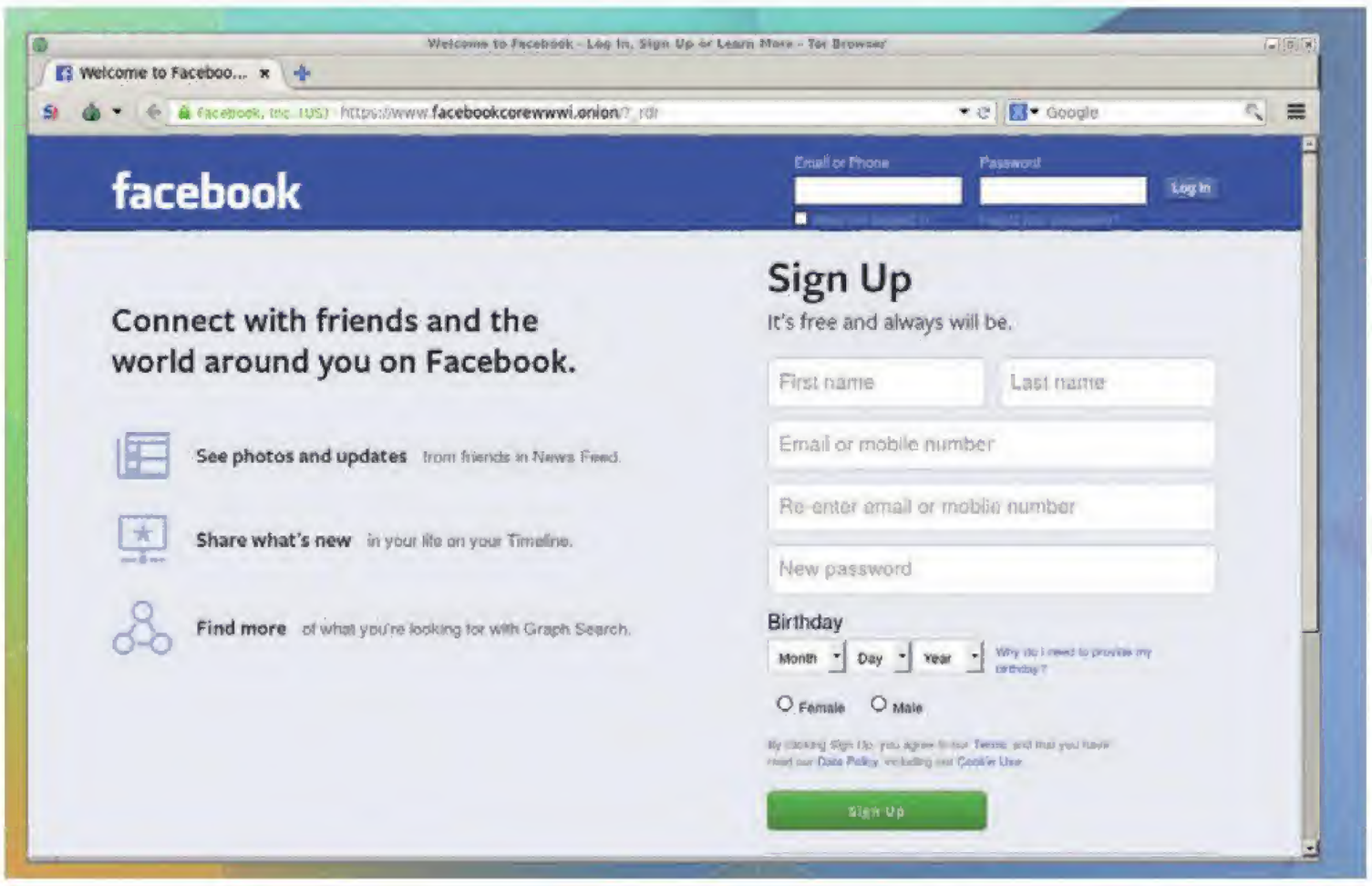
Certificates

All encrypted communication requires some form of shared information to start. This could be a passcode that both parties know or an encryption key. In the case of HTTPS, it's certificates. These certificates include a public key for the organisation, and some information about how to use the certificate (what organisation it's valid for, what dates it's valid for etc).

When you install a web browser, it comes with some certificates installed by default. These are root certificates, and the browser trusts the organisations that issued them completely not just to encrypt traffic, but to verify other certificates. When you visit a HTTPS website, the web server sends a certificate that has been cryptographically signed by a certificate authority. If the signature on this certificate matches one of the root certificates in your browser, then the page is accepted as valid.

This means that the entire basis for the security of HTTPS lies in these root certificates. If some malicious party manages to get the private key to one, they can break every bit of security in HTTPS. This also means that if someone can install a new root certificate on your computer, they have complete control over your web traffic. Many companies install root certificates on employees' browsers to allow the organisation to monitor and control internet activity.

Security settings can be confusing for non-technical people, so the *Tor Browser* makes things simple with a slider.



As well as letting you browse the web anonymously, *Tor* lets you host web pages anonymously using **.onion** domains. Here's Facebook served from **https://www.facebookcorewwwi.onion**.

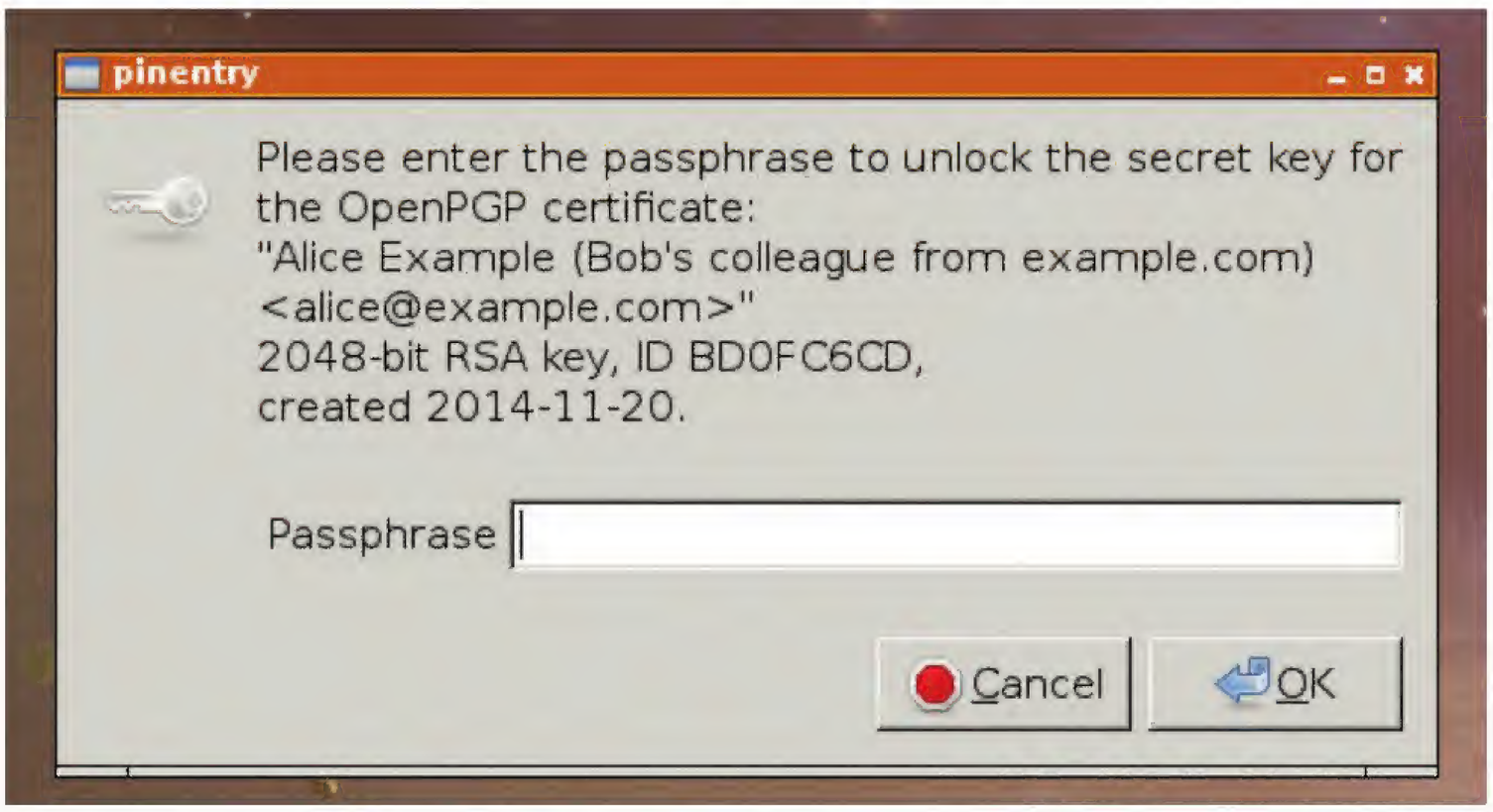
Communications

How to keep your online chats private.

The internet is about far more than just browsing the web, and the most important area for privacy on the net is online communications. There's a good reason that you put letters in envelopes in real life – you don't want everyone reading your mail. In the digital world you should ensure the same level of privacy by using strong encryption.

Email is still one of the most common forms of digital communication; however, it has no security built in. None. By default, there's no attempt to encrypt the communication, and no attempt to even verify that the person sending the message is really who they say they are. Over time, some solutions to these problems have emerged, but they're not universally applied. When sending or receiving an email, you should assume that there's no security at all.

When using webmail, bear in mind that many webmail providers make their money through advertising and may be mining your mail for information about you that can be used to better sell advertisements to you. Therefore, the first thing you need to do if you want private email is to use a mail provider that's not spying on your mail. This means not using an advertising-driven mail provider. **Riseup.net** is a good option. Another is to host your own email server, though this can be a little involved. You should be wary of any email provider that makes exaggerated claims about the total privacy of their system since, this isn't possible using the current email setup unless you use end-to-end encryption.



For more details on GPG encryption, see our masterclass in Linux Voice issue 12.

Many email platforms offer encryption to the server. On web-based email this is a HTTPS web page; on a regular server, this will be something like STARTTLS. This is an essential bit of encryption, because without it, the email is readable by anyone. However, alone, it doesn't offer any guarantees of privacy because the mail server could be reading the email, and it could send it unencrypted to the recipient's mail server. End-to-end encryption is needed to ensure privacy. This means that you need to encrypt it yourself before you send it, and this needs to be done in such a way that only the person receiving it can decrypt it. The standard method for this is *Gnu Privacy Guard* (*GPG*). This can be used in two ways: encryption and signing. Encryption means that only the intended recipient can read the email, while signing means that anyone can read it but it guarantees that the mail came from the person that signed it.

GPG uses public-key encryption for verification of identification and key-exchange, symmetric encryption for privacy and hashing for signing. In order to use *GPG* you have to create your own public key, and get the public key of anyone you wish to communicate privately with. You can either do this by exchanging key files in person, or by using a key server.

Thank GNU for privacy

The method of setting up *GPG* varies significantly depending on what mail client and mail server you're using. Unfortunately, there isn't yet a simple solution that works across the board. You should look up the advice for your setup on the mail client's website. When properly set up, *GPG* protects the contents of the message, but doesn't hide who is communicating with whom. This, and other metadata stored in the email header, may still be sent in plain text.

While there's no easy way of hiding the metadata in an email (or a good alternative that can be guaranteed to be secure), there are some options to mitigate the problem. You can completely hide your location by accessing webmail through *Tor*. This means that it's impossible to link the email to the physical location sending it. If you do this, and use different email addresses for different things, you can achieve a reasonable level of anonymity even though the metadata is still public.

Next generation private communications

All the methods of communication we've looked at in the main text are client–server. That means that your messages are first sent to some central server, and then on to the intended recipient. They can be secured through end-to-end encryption, but it's hard (or even impossible) to protect metadata, and potentially, an encrypted service could be forced offline by an overzealous government that wants to limit the options for secure communications (as happened with the Lavabit email service).

The alternative is a peer-to-peer setup, similar to how BitTorrent works. A service like this would be impossible to shut down. At present, there isn't a widely-used peer-to-peer chat system, but there are a couple in development.

- **Ricochet (https://github.com/ricochet-im/ricochet)** uses *Tor*, and each peer has its own hidden service as its interface with the network. This provides a strong degree of anonymity (though not perfect as law enforcement agencies have been able to de-anonymise hidden services in the past).
- **Tox (tox.im)** focuses less on anonymity, and more on having a robust network that's hard to shut down, and on secure encryption.

While both these projects are potentially very valuable assets in the fight for privacy, at present we can't recommend either of them for secure communications because they are simply too immature. They're under rapid development, and that could lead to bugs. However, in the future, when they settle down, they may provide good alternatives to the traditional client–server tools.

While email is still hugely popular, instant messaging (IM) can be more convenient. Like email, there's often little security built into IM solutions by default, and many IM platforms are run by advertising companies that mine the chat sessions for data. Many proprietary IM platforms make claims about privacy and security, but are very hazy on the details.

Getting chatty

For privacy, you need end-to-end encryption, not just encrypted communications to the control server. Off The Record (OTR) is a layer of end-to-end encryption that runs on top of an IM session to provide privacy. It can run on top of any instant messaging platform, but the developers of the Tails distro recommend that it's only used with IRC and Jabber (or other XMPP platform). OTR is a plugin for *Pidgin*, and you can download the source code or Windows binaries from **https://otr.cypherpunks.ca**. It's in most distros' repositories, but make sure that you have the latest version (check the OTR website for up-to-date details).

"Like email, there's often little security built into instant messaging solutions by default."

Another option is to use OTR and *Pidgin* through the Tails live distro. This is a good option if you plan to use OTR through *Tor*, since using Tails will ensure that everything is set up correctly. There are details of the Tails OTR setup at **https://tails.boum.org/doc/anonymous_internet/pidgin/index.en.html**.

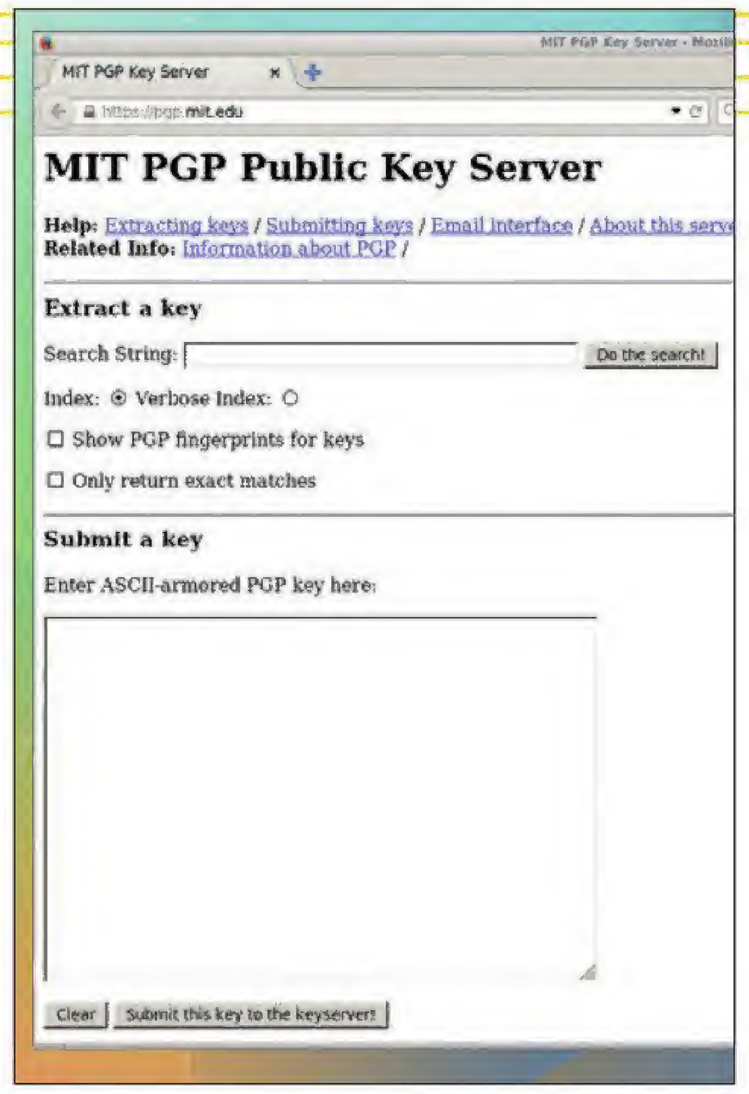
Both parties in the communication need to have OTR installed for it to work (if you try to initiate an OTR session with someone who doesn't have it installed, they'll get a message telling them how to install it).

The first time you chat with someone, you need to make sure they are who they say they are. OTR offers three different methods of authentication:

- **Shared secret** Using this method, both users see a text box and have to type in some text. If they both enter the same text, they're authenticated with each other.
- **Question and answer** One user poses a question to the other, and enters what they think the answer should be. If the other person enters the same answer, then they are authenticated.
- **Fingerprint** Each user has a hexadecimal string linked to their username that's known as a fingerprint. They can share this string with other people either when they meet them in real life, or by some other means of secure communication. OTR displays both users' fingerprints, and if they match what the users are expecting, they can authenticate each other.

The first two can be used to authenticate someone you know, and don't rely on you being able to exchange cryptographic keys in any way. They just need you to be able to come up with something that you'll both know. The final method can be used if you've already exchanged digital fingerprints.

OTR isn't anonymous, and people can still see who you've communicated with. However, the messages are designed in such a way that even though a spy can see that a message has been sent, they can't verify that it was signed by a particular public key, and there's a tool to generate fake messages (ie messages that appear real to a spy, but are garbage to anyone involved in the chat). This means that, while it's not truly anonymous, there is some deniability, since no-one except the intended recipient can prove a particular message was a real message sent by you and not a fake.



Key servers are an important part of using GPG effectively. You can add your keys to pgp.mit.edu (one of the most popular) via their website.

On the move

It's not just messages sent via the internet that are routinely intercepted: phone communications are too. Both voice and SMS messages are sent unsecured and are intercepted by phone companies. Our recommendation for private communication on the go are the tools by Open Whisper Systems (**https://whispersystems.org**). These include Text Secure (an encrypted mobile instant messaging platform) and RedPhone (an encrypted voice caller). Both of these are available through the Google Play store and iTunes.

An added advantage is that both of these apps are free to install and use and neither comes with advertising. Instead, the software is funded by grants from privacy advocates such as the Freedom of the Press Foundation and the Shuttleworth Foundation (as in Mark Shuttleworth, the Self-Appointed Benevolent Dictator For Life of the Ubuntu Foundation). It's not just us recommending these tools. They come with an endorsement from Edward Snowden himself who said, "Use anything by Open Whisper Systems." LV

In September 2013, *Mailpile* crowdfunded over $160,000 to develop a webmail client that makes GPG encryption simple. It's still in beta, but we expect a 1.0 release later this year.

LINUX VOICE PROFIT DONATION SCHEME 2015: THE WINNERS!

Thanks to you, Linux Voice readers, we are able to help FOSS-related projects and organisations. Here's how it happened.

When we decided to create Linux Voice back in late 2013, one thing was very clear right from the start: we should give something back. We all get so much from the Free Software world – not just great software, but an awesome community willing to help and spread the word.

So we decided to do two things. First, as you'll have seen if you're a frequent visitor to our website, we've been making back issues available under a Creative Commons licence nine months after they go off sale. This means that anyone can read, share and modify our old content – and even sell it on – providing they give Linux Voice credit for creating it in the first place. We've already seen articles translated and updated to match the latest developments in FOSS, so this is working really well.

"We decided to give 50% of our year-end profits back to Free Software projects."

Second, we decided to give 50% of our year-end profits back to FOSS projects, communities and organisations, and let our readers choose exactly where the money goes. We asked our website visitors to come up with a shortlist of candidates, and then in issue 13 we started the voting procedure. You might be wondering: why did it take so long? Well, we wanted to make sure all readers had the chance to vote, and it takes a while for the magazine to appear around the world.)

As we've spent the last 12 months getting the magazine established, sorting out printing and distribution, we don't have a giant pot of money to give away. But we have £3,000 to contribute, and to make sure multiple projects benefited, we split the winners up into two categories. The first is software projects, while the second is distros and organisations. So without further ado...

SOFTWARE

Desktop publishing application *Scribus* won by a considerable margin, which isn't surprising as we'd like to make the whole magazine using it some day. It's an excellent program and is being used by many professionals around the world; see our report in issue 15 for the features we need before we can make the switch. Craig Bradney, one of the *Scribus*'s lead developers, told us:

"I'd like to thank all the readers of Linux Voice magazine, as well as all of the users of *Scribus* out there. These kind of funds go well towards allowing the team to meet up and collaborate on features and bug resolution. The best example is the Libre Graphics Meeting which has just happened in Toronto, which was the 10th anniversary of LGM, where *Scribus* has been represented every year and we've always had a good meeting there. By the time this news hits, out first development version of the 1.5.x series will have been released. We hope the readers enjoy the new development version and all its new features. Thanks again!"

Gimp and *Inkscape* are also valuable tools that we use in the production of Linux Voice, so hopefully our cash injection can help those projects to add new features and close bugs as well.

1st Place Scribus receives £1,000
2nd Place Gimp receives £300
3rd Place Inkscape receives £200

PROJECTS AND DISTROS

The Open Rights Group (ORG) and Electronic Frontier Foundation (EFF) have similar goals: they are organisations working to preserve digital rights and freedoms, in an age where governments are using mass surveillance programmes to spy on us all. We regard the internet as one of the most important developments in human history, a fantastic way to share knowledge and ideas, and it's vital that it doesn't become locked down or massively restricted.

The Open Rights Group is based in the UK, and Jim Killock, its executive director, told us: "We are really grateful to all of the readers who voted for ORG. We're going to use the money to print our report into surveillance by GCHQ [Government Communications Headquarters] and send copies to every new member of parliament so that they can understand how privacy, free speech and the security of the Internet is being threatened by our own government agencies. This will give us an early start in challenging any new intrusive legislation proposed by the next government. Thank you!"

This is especially good now that the UK has a new government, and has already expressed a desire to expand its mass surveillance operations. Hopefully the Open Rights Group can show MPs that we can fight for our security and protect ourself from terrorism without having to throw away all our civil liberties.

So, thanks to everyone who voted, and above all, thanks to you, whether you're a long-time Linux Voice subscriber or this is the first copy you've grabbed from a newsagent. Your support for us has made this possible, and we're really happy to give something back. Tell all your friends to subscribe and next year we can contribute to even more projects! LV

1st Place Open Rights Group receives £1,000
2nd Place Electronic Frontier Foundation receives £300
3rd Place Free Software Foundation receives £200

THE OLD MEETS THE NEW

Two decades of development have brought SUSE more than just Yast and a slick KDE desktop. We visit the distro's team in Nuremberg to find out more.

Back in the late 90s, as Linux was starting to gain mindshare, distro boxed sets were all the rage. Most internet users still had sluggish dial-up modem connections, so it felt like Christmas when a chunky package crammed with Linux CDs and manuals arrived at the doorstep. And SUSE Linux (**www.suse.com**) was the best in this regard: its documentation was excellent, the boxes included thousands of packages spread across multiple CDs or DVDs, and the distro had an overall feeling of quality and refinement to it – German precision engineering, you might say.

"SUSE has always had a feeling of refinement – precision German engineering, you might say."

Since then, a lot has happened with SUSE. The company was bought up by Novell, then Attachmate, and is now an independent business unit under Micro Focus International. Home desktop users and hobbyists are most likely familiar with OpenSUSE, a community supported distro, while SUSE Linux Enterprise (SLE) targets big business and provides competition for Red Hat Enterprise Linux (RHEL).

At the time of writing, SUSE was hiring 73 new staff, and business appears to be booming. But we also don't hear much from the distro team. Everyone knows that Canonical is doing flashy stuff with Ubuntu on phones, while Red Hat demonstrates the most bleeding-edge Linux technology in Fedora. So what's SUSE doing? Is it really a conservative company that doesn't have much to shout about? Or is there more happening behind the scenes?

We paid a visit to SUSE's offices in the Franconian capital of Nuremberg, home to around 200 staff including developers for both the OpenSUSE and SUSE Linux Enterprise distros. Before we went, we asked Linux Voice readers on our website what questions we should put to the team, so read on to find out what's next for SUSE...

OpenSUSE vs SLE vs Tumbleweed

How do the different distros work together?

One thing many of our readers wanted to know is: what is the relationship between OpenSUSE, SUSE Linux Enterprise, and the Tumbleweed rolling-release distro? To find out, we talked to Douglas DeMaio and Richard Brown. Douglas is the only SUSE employee who works on OpenSUSE full-time, and explained his role:

"I'm the OpenSUSE employee, the one and only, so I basically do PR, marketing and a variety of other things from a business outlook, so we can coordinate efforts and do things properly – save money where it is appropriate. I look at the overall goal and how we want to project OpenSUSE going forward."

Richard Brown is more directly involved with the technology in the distro: "I have two jobs. Four days a week I'm a QA engineer working on SLE, and eight hours a week I'm chairman of the OpenSUSE board. The board leads the project, and I look after the board. The rest of the board is elected by the community, and it has five elected seats. The charter forbids any company from having a majority – so there can only be two other SUSE employees on the board. The rest have to be external community members."

Richard explained that this is different to other projects which have more hierarchical decision making models, such as Fedora. But what happens with major changes like the switch to *Systemd*? Was there a big conflict and a vote, like with Debian?

"No! We had some people who were willing to make the switch, and nobody who was not willing to do it. So we just did it. There was a little bit of a user backlash, but that was easy to explain – we could show that *Systemd* actually worked. So the model works really well for us, but our board is quite different to that of Fedora with all its structure and special interest groups and governance."

We asked Douglas and Richard to clarify the position between the various SUSE distros. Recently, the company made available the source code for its SUSE Linux Enterprise distro, and that has been integrated into the Open Build Service (OBS), a system used for building SUSE (and other distro) releases.

"The entire SLE codebase is now in OBS for OpenSUSE to build upon. The relationship and role of the different distributions inside SUSE is changing – and where it's going to end up, we're not sure yet, as that's the fun part of this stuff! The community is shaping it as we speak, and it's going to happen organically. There's only one full-time OpenSUSE employee, and there are a few other SUSE employees that have very OpenSUSE-centric roles. OpenSUSE is very much its own thing, it's very independent – much more so than comparable projects. The community can really decide where it wants to go. Like, with KDE being the default desktop, while SLE doesn't have KDE at all – it's Gnome-only in SLE 12."

Pick-n-mix

Richard describes how parts of OpenSUSE are making their way into SLE more often now, and in a more diverse way. And indeed, some parts of the Tumbleweed distro are being used in the enterprise product – which seems odd, when enterprise distros are usually very conservative and rolling releases are typically bleeding edge. For instance, OpenSUSE didn't include support for the Btrfs filesystem by default, but the SUSE team had tested it in Tumbleweed and found it ready for widespread use, so they included it in SUSE Linux Enterprise 12.

But now that the SLE sources are available to everyone, does SUSE expect the community to create rebuilds, like CentOS and Scientific Linux did with the RHEL sources?

Richard: "We want to do it very differently from that. We realise that might happen, but we think we can do something way far exciting than that. Basically, we want to redefine the OpenSUSE regular release as effectively a hybrid, with the SLE base underlying it, and community stuff on top. So you have a very stable SLE-like or CentOS-like base system, up to and

Bottom left Around 200 staff work for SUSE in Nuremberg, primarily on the enterprise products but partially on OpenSUSE too. (Photos: Alyssa Mello)
Bottom right SUSE's office includes a mini museum with boxed sets dating back to the very earliest releases.

SUSE people showed us the latest updates to the Open Build Service and OpenQA projects.

including minimal X [graphical layer], but anything the community wants to maintain at a faster pace, we can put on top. From the community perspective we think that's more exciting than something like CentOS, which is simply an exact copy of RHEL, and also from the SUSE side; SUSE the company is more interested in the outcome of that because anything that's built in that way is a candidate to be included in the next SLE service pack."

Mixing components in an enterprise-level distro is difficult though. On the one hand, users expect an extremely solid base system that doesn't randomly change underneath you and break all of your applications. On the other hand, software moves at such a rapid pace that you need regular updates to some things, and not stick with the same versions for the whole 10-year support period. In SLE 12, SUSE introduced "modules", which are parts of the distro that move at a faster pace than the normal packages. They're distributed and supported by SUSE, but not to the same level – in other words, you can't phone up and get a bug fixed within a guaranteed timeframe.

"In SLE 12, SUSE introduced modules, which are parts of the distro that move at a faster pace."

Still, even with this approach to mixing elements of SLE and OpenSUSE, some people will simply want a carbon copy of SLE without the commercial support requirements. The SUSE team doesn't expect to build such a distro themselves, but can envisage the community doing it.

Tumbleweed: the future?

And what about Tumbleweed? Will it become the standard distro that SUSE expects hobbyists and home desktop users to run?

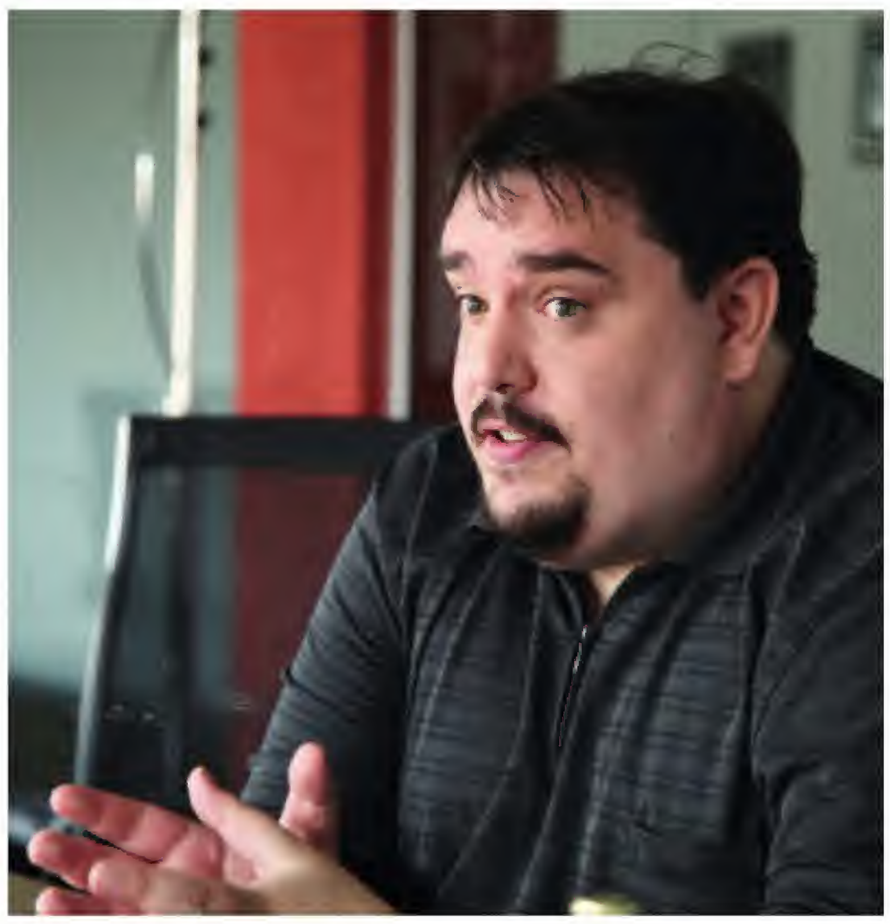
Richard: "In my opinion, rolling releases are the future of Linux development. It fits in with all the trends we're seeing everywhere else with devops and continuous integration. It's the Linux equivalent of that. You see it in the kernel, and the model works. We've managed to build a rolling distro that works to that level – that people use every day. So for those users who want something that's stable, usable, but also always getting the latest stuff, which we think covers the enthusiast crowd, Tumbleweed is it."

But what about breakage? Despite the best efforts of the Arch team, for instance, major updates occasionally break things and users are expected to keep an eye on the distro's news site and wiki. Moreover, Arch users tend to be more technically inclined and can fix issues manually. So how does the SUSE team prevent breakages in Tumbleweed from ruining the experience for general hobbyist users?

Advantage: SUSE

"We have secret sauce that Arch doesn't have! We're a very tool-centric distribution, and we've always thought in those terms. From pretty much day one when OpenSUSE started 10 years ago, we started the Open Build Service, because we needed some system to build our distribution in a very open way. We made it very cross-distro. Now we have OpenQA, our automated testing tool. In the case of Tumbleweed it's totally paired in with the development process and also the Open Build Service. So when we're building something new for Tumbleweed, before it gets anywhere near Tumbleweed, it gets tested."

Richard went on to describe the case with *GCC 5*. It's a big update to one of the most critical components of the system, and it's currently sitting in a staging area. With the Open Build Service and OpenQA, the SUSE team can constantly check how much of Tumbleweed can be built correctly with *GCC 5*, and what packages are still broken. The same goes for kernel updates and other major changes – they can't be released to Tumbleweed users until they've passed OpenQA's automated tests.



Richard Brown is chairman of the OpenSUSE board, and works on QA for the distro.

Beyond the distro

Open Build Service and OpenQA: tools for all distro makers.

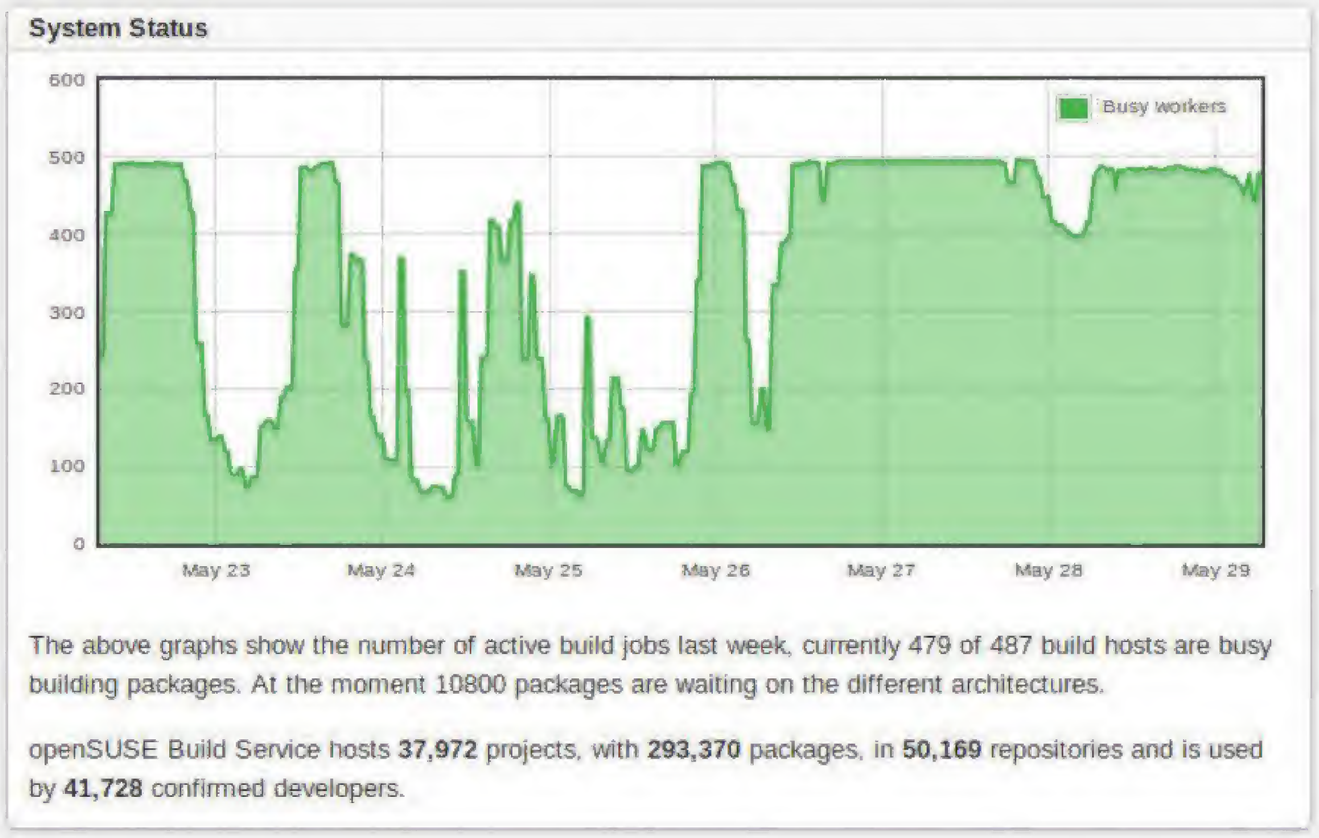
Constructing a distro from the ground up is hard work, as you'll know if you've ever tried Linux From Scratch. The more processes you can automate, the better. For building OpenSUSE and SLE, the SUSE team uses the Open Build Service (OBS, **http://openbuildservice.org**), an open source project that was originally created just for SUSE projects but is used by many more around the world. OBS can automatically generate distro ISOs from packages and scripts, and upload them for testing.

Also, with the OBS you can create packages for many different distros. Right now, third-party software distribution on Linux is rather messy: if you've written an awesome piece of software, how do you get it to Linux users? You could wait until some distro developers package it up, but that could take many weeks or months before your program is in Debian, Arch, Fedora and so forth. Or you could try to hack together a statically linked binary, wrap it up in a tarball and throw it onto your website – but that's not the most elegant solution.

With the Open Build Service, you can upload your code and various metadata (eg for dependencies), and then build binary packages for specific distros. This means that end users can download packages that have been compiled for their own distros, without having to do a lot of manual work extracting tarballs. And this isn't just a service for part-time coders packaging up their latest GitHub work – it's used by major projects as well.

Go to the OwnCloud website (**www.owncloud.org**), for instance, and then the Downloads section. If you choose Linux packages, you'll be redirected to the OwnCloud section of the Open Build Service website. Many developers build their packages on SUSE's Open Build Service machines (the reference servers), but as OBS is open source, it's possible for companies to set up their own servers and perform package builds internally.

Along with packaging, another vital part of distro development is testing. Again, if you can automate this process as much as possible, you can save a huge amount of time. SUSE's OpenQA suite (**https://openqa.opensuse.org**) lets you boot up distros in virtual machines, send them virtual keypresses and mouse clicks, and see how they respond. In this way, you can rebuild your distro every night, for instance, then boot it up in a virtual machine and test various functions of it.



The Open Build Service reference server hosts a whopping 37,972 projects and over 290,000 packages.

Automated testing with OpenQA

But how does this work? OpenQA does some nifty tricks like screen-scraping – that is, pulling in images from the virtual screen, using optical character recognition to capture some text, and comparing it against an expected result. The SUSE team showed us this in action, with an OpenQA test suite that booted up a desktop Linux distro, attempted to open a terminal, and echo some text to the screen. OpenQA can find this text (even if it's in a slightly different place due to desktop theming changes), or if it doesn't appear, register a problem for a developer to fix.

So by creating an OpenQA test suite you can automatically test your distro in many ways: check that it installs OK, try more complicated partitioning, run all major apps after installation, and so forth. While OpenQA started as a SUSE project, the team showed us how other distros are now using it, and it's even possible to perform checks on non-Linux platforms such as Windows.

The notion of "eating your own dogfood" (that is, actually using the software you develop in your daily work) is integral to the open source world, so it's good to see the SUSE team using OBS and OpenQA extensively. And kudos to the company for supporting the spirit of open source by making these tools available under the GPL and helping other distributions utilise them. LV

Yes, OpenSUSE even has its own beer – get a taste of it by heading to the distro's next conference.

INSIDE THE LINUX KERNEL

Since he couldn't find the blueprints for Atlantis, **Mayank Sharma** looked at the next best thing – the Linux kernel.

It was the turn of the century but Allen Pais couldn't get to the graphical desktop on his computer. That was because Red Hat Linux 6.0 didn't support his SiS 6215c graphics card. Instead of moaning, he decided to get his hands dirty, read the Linux kernel source and the kernel mailing list archives and hacked together a driver. Although his driver didn't work, the experience gave him wonderful exposure to the kernel as well as a career – Pais is now the principal kernel engineer at Oracle.

Pais is just one of the thousands of contributors to what's been dubbed the world's largest collaborative development project. According to estimates it would cost billions of pounds and thousand of man years to redevelop the kernel, which now contains over 19,000,000 lines of code written in about a dozen programming languages. Most of the work is paid for by over a thousand multinational corporations that have developers like Pais to work on the kernel.

"It would cost billions of pounds and thousands of man years to redevelop the kernel."

The secret sauce behind the world's largest open source project is an effective system of collaboration. Unlike other open source communities, the kernel community has had to evolve its own distinct mechanism of operating in an environment with thousands of eyeballs and where thousands of lines of code are modified every day. This mechanism enables contributors like Arun Raghavan, who has had just one patch accepted to the kernel, to improve the kernel with their "well-meaning ignorance". Raghavan, who is a developer on the *GStreamer* multimedia framework and maintains the *PulseAudio* audio server, has sent in patches related to issues with his Macbook Pro that weren't accepted but managed to kickstart a discussion towards the proper fix.

If you're not a developer and have never contributed to an open source project, think of a patch as a record of changes to an existing piece of code. When developers need to fix a bug or add a new feature to the kernel, they write up a patch containing the list of changes they want to make to the kernel either by replacing lines of code or adding new ones or both.

The patches go through an elaborate vetting process before making their way into the kernel. Once a patch is submitted, other developers review it for quality and whether, in fact, the change it implements is something they want in the kernel. If the change is minor and implemented nicely, it's

accepted without much delay, while others can linger for years. Raghavan has had experience with both: "The first security patch I posted was a minor cleanup that I thought made the code more consistent. The maintainer felt that this introduced a bit more overhead. That discussion just fizzled out. The second was incorrect, so wasn't merged. And the third was the minor API change that did get pulled [included], and was really quite painless."

Tending the garden

A typical release gets over 10,000 patches, so it's just impossible for all of them to be inspected by a single individual. Instead the kernel is segregated into several logical subsystems or trees such as networking, video drivers, etc. Each subsystem tree has a specific maintainer in charge of approving patches for their particular subsystem. These maintainers each manage their own version of the kernel source tree.

Once a patch is submitted for review, the developer will receive all sorts of feedback on the submission and should be prepared to make changes to their code accordingly. For many developers, the review process is one of the most intimidating parts of the kernel development process. "It took a lot of courage to send the first patch," says Pais. "I had to ensure I read every piece of documentation about how it has to be sent."

Despite his apprehensions Pais has had a wonderful experience working with the developers who have been very helpful even when he's made mistakes, like the time his patch broke the kernel. "It

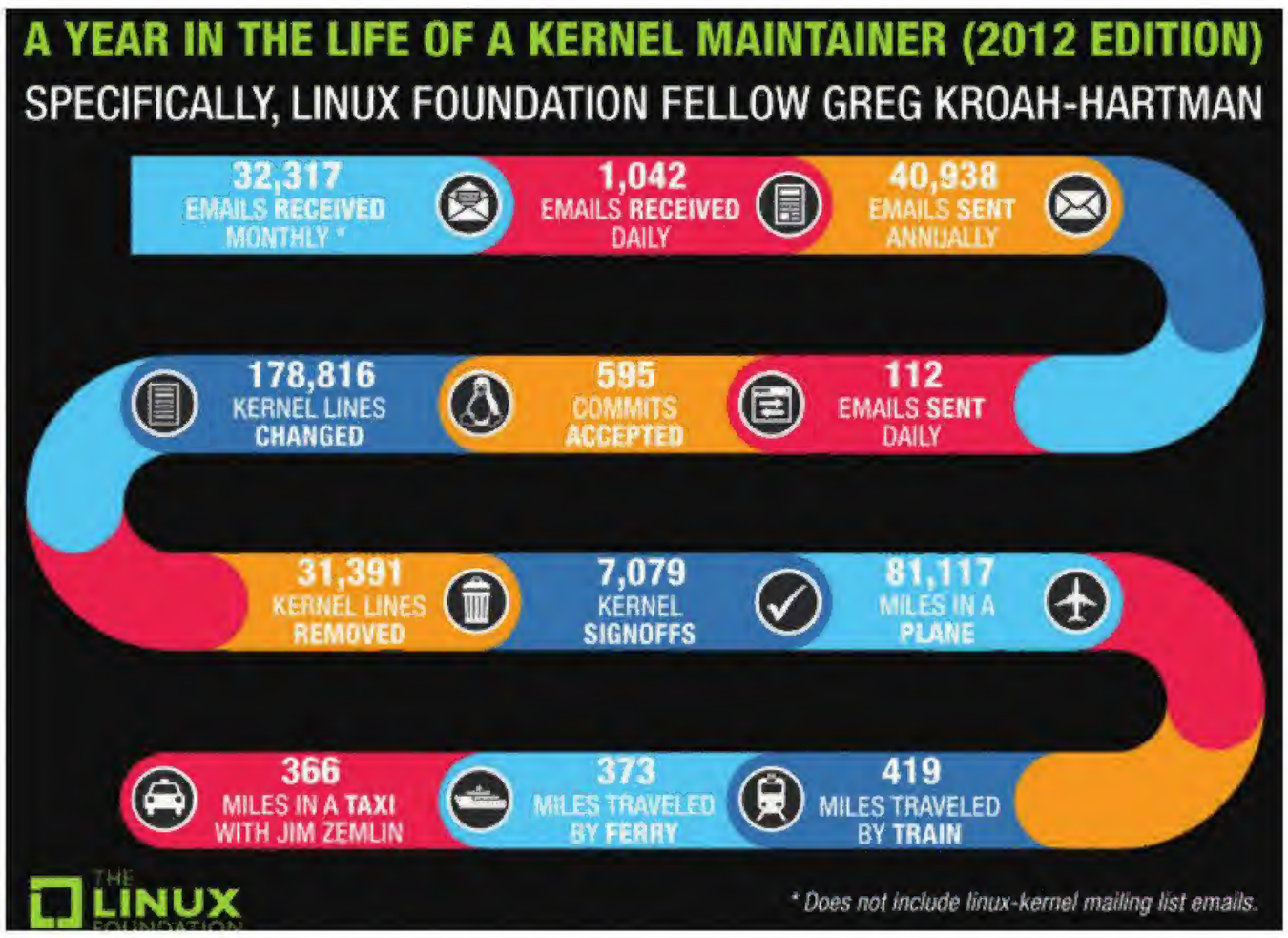
Top left Arun Raghavan often reminisces about his time as a Gentoo developer.
Bottom left In addition to the kernel patches they receive, Josh Boyer values the help from the community in triaging bugs which helps Fedora as a whole.
Above right Jonathan Corbet's guide to the kernel development process is a wonderful read for any developer who wants to get started on the Linux kernel.
Image Credit: Linux Foundation, Flickr

Linus's lieutenants

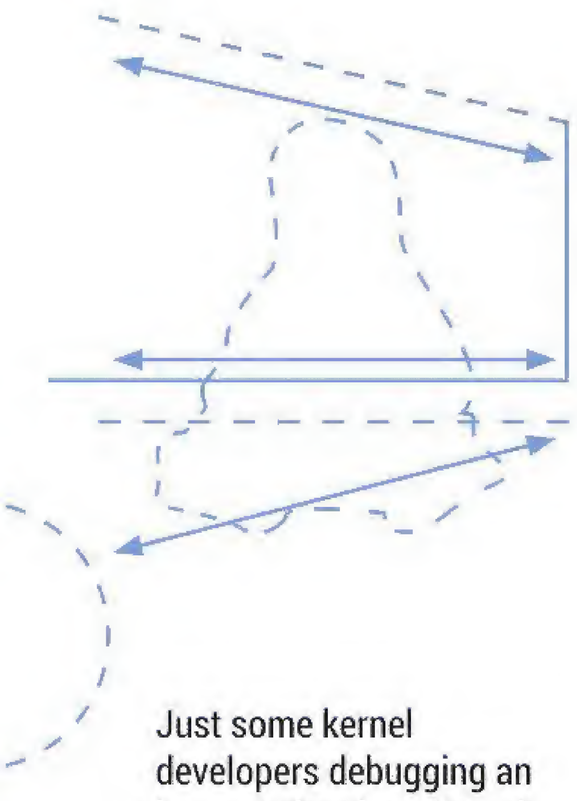
The 2015 edition of the Linux Kernel Development Report by the Linux Foundation gives a wonderful overview of the kernel development process. The report tracks the development of the Linux kernel during a specific period and analyses various factors such as the developers doing the work and the companies sponsoring them.

According to the report, since the 2.6.11 release in 2005 more than 11,000 developers have contributed to the kernel. However, despite the large number of individual developers, most of the work is done by a handful of developers. The report also notes that the total number of patches signed off by Linus Torvalds (329, or 0.4% of the total) continues to decline. "That reflects the increasing amount of delegation to subsystem maintainers who do the bulk of the patch review and merging." Greg Kroah-Hartman tops the table by signing off 13,028 patches (14.4%) that he didn't author himself, followed by David Miller (8.6%), Mark Brown (4.1%) and Andrew Morton (4.1%).

Ranking sponsored contributions by companies puts Intel (10.5%) at the top followed by Red Hat (8.4%) and Linaro (5.6%), while 12.4% of the contributions come from developers who aren't paid for their time and contribution.



A year in the life of veteran kernel developer Greg Kroah-Hartman.
Image Credit: Linux Foundation



Just some kernel developers debugging an issue at the Linux Kernel Summit in 2008.
Image Credit: Jonathan Corbet, LWN.net

was a learning experience and fortunately people from the Linux community have been really nice."

Once a maintainer approves a patch, it's entered into their subsystem tree. A maintainer accepts a patch by adding a 'signed-off-by' line to the code. Some maintainers might have multiple trees; one for an upcoming kernel release and another for a future release, for example. After a patch is included in a subsystem tree, it's bound to get more eyeballs as it'll now attract the attention of developers who are working on that subsystem's tree.

But what happens when, say, a patch uses a function that's been changed by another patch? To avoid such conflicts and to enable developers to preview all of the patches being prepared for the next kernel release, the Linux kernel uses staging trees. These trees are a collection of the patches that come in from the various subsystems.

In a blog post, kernel developer Greg Kroah-Hartman explains that the staging tree "is used to hold standalone drivers and filesystems that are not ready to be merged into the main portion of the Linux kernel tree at this point in time for various technical reasons." The staging tree resolves "the 'hundreds of different download sites' problem that most out-of-tree drivers have had in the past" and gives developers a singular place to concentrate their efforts.

Releasing a kernel

Since 2005, the Linux kernel has switched to using a merge window to incorporate patches into the kernel. During this merge window Linus Torvalds accepts patches into his kernel tree, which will eventually be released as the next version of the kernel. The merge window lasts for two weeks. When it ends, Torvalds releases the first of the -rc kernels, such as 4.0.2-rc1.

Instead of reviewing each and every patch, Torvalds trusts the maintainers to only send quality patches for merging into his kernel. Before it is released, Torvalds's -rc kernel is put through rigorous testing. Patches that causes a feature to stop functioning, which is known as regression in software development, are the first ones to get the axe if they aren't fixed immediately.

As the kernel releases go through a period of testing and stabilisation, Torvalds puts out new -rc releases once every week. A typical kernel release cycle has anywhere between six to nine of these -rc releases. Once it has been through testing, this kernel is released as a new version and is dubbed the mainline kernel. The whole process usually takes about 10 weeks. The actual time between kernel releases tends to vary a bit, depending on the size of the release and the length of time it takes to weed out any bugs.

In the current scheme of things, this kernel release containing the latest features and fixes isn't considered a stable release as it hasn't been tested long enough. Every release also has a corresponding "stable" release which contains just the security updates and bugfixes. There might not be noticeable changes in a new kernel version, unless it adds support for a piece of hardware that you use. You can view a summary of the differences introduced in each new kernel on the KernelNewbies website (**http://kernelnewbies.org/LinuxChanges**).

While all Linux distributions have the Linux kernel at their core, it's very unusual for the major distros to ship the upstream mainline kernel as is. To wrap our

Strip searching the kernel

One noteworthy user of the mainline kernel is the Linux-libre project. This Free Software Foundation-sponsored project strips the kernel of everything "that is included without source code, with obfuscated or obscured source code, under non-Free Software licences, that do not permit you to change the software so that it does what you wish, and that induces or requires you to install additional pieces of non-Free Software."

According to the project the kernel started to include binary blobs in 1996. The project uses scripts to check and remove proprietary firmware from the kernel and produce a 100% free software version. This kernel powers several free software distros, including the FSF-endorsed Trisquel.

Compile your own kernel

Although the kernel is the core component of a distribution, you can replace it with a custom one without much effort. Replacing the kernel with a custom one is a wonderful way to get a peek at how it works. It's also one of the first things you should get a grip on if you wish to contribute to kernel development. Compiling a custom kernel gives you the ability to tweak the stock kernel in a particular way, for example to enable an experimental feature.

Another reason for using a custom kernel is if you have hardware that isn't supported by your distro's stock kernel but is supported in the upstream kernel or a third-party kernel. The process also gives you access to the various compilation flags using which you can optimise it for your needs by stripping away any excess.

You can find instructions for compiling a custom kernel for your distro in its official documentation or on its wiki.

According to the Linux Foundation report, the FOSS Outreach Program for Women introduced 24 new developers who contributed 1.5% changes in the run up to the 3.18 release.
Image Credit: Linux Foundation, Flickr

heads around the work that distros do to incorporate Torvalds's mainline kernel into a release we reached out to Josh Boyer, who is one of the three members of the Fedora Kernel team.

How distros use the kernel

Fedora adopts two different approaches to selecting the kernel for a release. The project's development branch, known as Rawhide, follows Torvalds' tree on a daily basis. "We build Git snapshots of his tree every day and it simply keeps following along," says Josh. On the other hand, the stable Fedora releases are based around the latest stable release of the kernel: "Fedora 20 and 21 are both on 3.19.8 at the moment, and Fedora 22 is on 4.0.2." During a release's lifetime, Fedora rebases the kernel, which is to say it swaps out the old kernel with the most recent stable release. This means that Fedora 21 will soon be rebased to the 4.0 branch, and Fedora 20 will follow shortly.

While Fedora, just like the other distros, applies patches to the upstream mainline kernel, it does its best to keep these to a minimum. According to Boyer, "the significant majority of the patches we carry are actually patches that are already headed upstream. They are either taken from the mailing lists or linux-next tree and backported to fix an issue we've found."

This approach has its benefits: "We've found that by staying as close to the latest upstream kernel version as possible, it is easier to work with the upstream developers. The code is still fresh in their minds, and they tend to be very responsive when we discuss issues with them. Using the newer kernel also brings in a significant number of bugfixes that our team wouldn't be able to scale to cover via backports."

That being said, there are always exceptions: "The most significant patchset we're carrying today is the Secure Boot work that we did in the Fedora 18 timeframe. We hope that will also eventually be merged, but it has been a slow process. We're also currently looking at kdbus and what we can possibly do to help the efforts there."

To summarise, Fedora 22 will ship the upstream 4.0.2 kernel release with 73 additional patches, of which 19 have already been merged in the upcoming 4.1 branch or one of the upstream maintainer trees. "10 patches are for additional ARMv7hl board support, 19 are for Secure Boot support and the rest are patches that provide simple default setting changes, or are in place to help us debug weird issues when they pop up." Put together, these patches will change 121 files: "The single largest addition is the Ethernet device driver we carry for X-Gene AArch64 boards, which is 10,320 lines by itself."

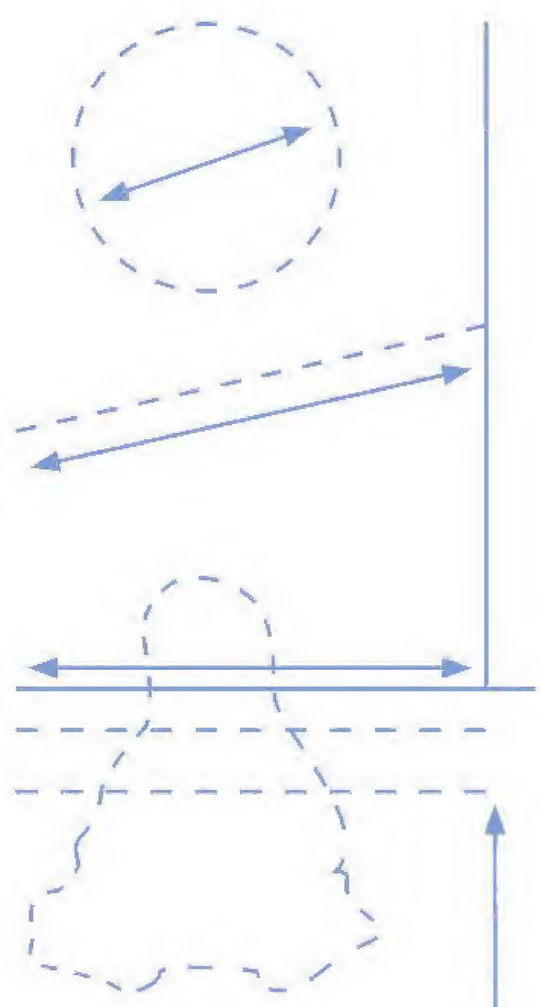
Boyer also shares the team's intention to increase contribution to the upstream kernel development: "While Red Hat employs a fair number of upstream kernel developers, Fedora is not the place where we are doing heavy kernel development. One of the goals for our small immediate team is to increase our upstream contributions, whether it be for bugfixes or small cleanups, or hardware enablement. We want to continue to be participants in the upstream kernel community and help where we can."

"From about 10,000 lines in 1991, the kernel has grown to over 19 million lines of code."

A massive undertaking

It is difficult for a non-engineer to fathom the size of the Linux kernel. From just about 10,000 lines in 1991 the kernel has grown to over 19 million lines. If you've collaborated on a developmental project, you can probably imagine the work and effort that goes into building and maintaining the most critical component of your distribution.

The Linux kernel is an engineering marvel, and one of its inflection points, as compared to the proprietary model of development, is community participation. The Linux kernel is one of the best documented open source projects, and contrary to popular belief, the kernel welcomes new contributors. In addition to the comments within the code there's plenty of documentation to welcome new developers. So budding contributors: grab your compilers, download a copy of the kernel and get cracking! LV



WRITE FOR LINUX VOICE

Linux Voice wants your ideas for tutorials, guides, how-tos and insights from the hacker world. If you've found something you want to tell the world about, let us know

What material is Linux Voice interested in?
Most of the time we're more interested in what you can do with software X, rather than singing the praises of software X itself. Clever software is good but useful software is better. Proprietary software that works on Linux is acceptable, but what we're most interested in is Free Software.

What don't you want?
We sometime get submissions that go like "I've been using Linux for X years; can I write for you?". This isn't very helpful, to us, because what we want to see is that you:

- Have an idea
- Can explain it clearly

If you can point us to examples of something you've written, please do – we're not looking for Shakespeare; we value clear communication and enthusiasm above all else.

What do you want?
Tutorials. We want tutorials, of around 3,300 words in length usually. We pay money! All tutorials should have a clearly stated aim, so readers know at first glance why they should follow it. "Get started with XX software" doesn't tell you anything; "Build a weather tracker with Python" is much more active and informative.

These are common reasons why we reject ideas:

- Something which has been covered repeatedly on Linux Voice and/or elsewhere
- Material not obviously related to Free Software
- Incoherent writing

Email ben@linuxvoice.com to write for Linux Voice

FAQ Qt

Its birthday gives us the perfect excuse to revisit a project used by almost everyone, from Canonical to KDE.

GRAHAM MORRISON

Q Hasn't the *Qt* toolkit been around for ages?

A *Qt* has indeed been around for ages. In fact, it's just celebrated its 20th anniversary, which is quite remarkable. We can't say the same for many open source projects. *Qt*'s first version, modestly called 0.90, was born in Norway on 20 May 1995, and was the culmination of four years work by its founders, Eirik Chambe-Eng and Haavard Nord. Both founders have since departed, and *Qt* itself has been through a few transitions to get here. In 2008 it was dramatically bought by Nokia, which was subsequently bought by Microsoft. It's now in the safe hands of a subsidiary of Digia called simply Qt Company – Digia was one of the largest users and contributors to the entire framework before buying *Qt*.

Q How is it pronounced again? Kew-tea?

A *Qt* is a serious contender in the open source silly names competition. Its developers and community pronounce it 'cute', which makes it feel a little like a hazing ritual for a developers' fraternity. Most other people unofficially call it 'Q T', which suits us fine.

Q The terms Toolkit and API are used to a lot in these pages, but what exactly are they?

A Both terms are almost synonymous, with toolkit becoming a more modern term for what mostly used to be called an API. Both help programmers to develop their software more efficiently. At their most basic, they're a library of functionality. For example, a programmer who wants to write something to calculate the square root of a number could research their own solution and write their own implementation. They could copy someone else's, or they could use a toolkit that offers the function along with lots of other related functions.

Q Surely there's more to *Qt* than calculating simple mathematical functions?

A Absolutely. Maths functions belong in a programmer's maths library. There's a function to calculate a square root in C's standard library, for instance, which is ubiquitous and as old as time(.h). But when you start creating libraries and grouping them together to help achieve a set of related goals, you have a toolkit.

Eirik Chambe-Eng and Haavard Nord started *Qt* after working on ultrasound equipment in a regional hospital in Trondheim, Norway, in 1990. They noticed how much extra effort it was taking programmers to port their work to different platforms and wanted to create a toolkit that would enable them to deploy their code on multiple systems with very little extra effort.

Q What features does *Qt* group together?

A If you want to create a toolkit that's going to work across different operating systems and environments, there's very little you can actually build upon that's common to them all. Input, output, sound, graphics – these all need to be unified within a toolkit so that the programmer doesn't need to worry about whether the user is on Windows, OS X or X.org. In an ideal world, the programmer wants to write code to add menus to their application and those menus would appear on whatever system the application was built for and running on.

> **"Each major revision of KDE has followed a major revision of the Qt toolkit."**

In reality, all the major operating systems use dramatically different systems and APIs for their own menu generation, and almost every other function they provide. And this is where *Qt* comes in. *Qt* is the bridge between these systems, covering everything the programmer might need – from user-interface design and layout, to remote procedure calls and accessing Bluetooth devices.

Q If *Qt* is so great, why isn't everyone using it?

A More projects seem to be using *Qt* than ever, but it could be akin to cracking a nut with a sledgehammer. If you just needed a library to split up character strings, for instance, it wouldn't make much sense installing *Qt* solely for its excellent string handling. It's a large download that would then become a large dependency. Incidentally, a huge area of improvement in *Qt 5* is its modularity, so that some parts of *Qt* can be installed without installing the whole thing.

Q With all this cross-platform love, what does this have to do with Linux specifically?

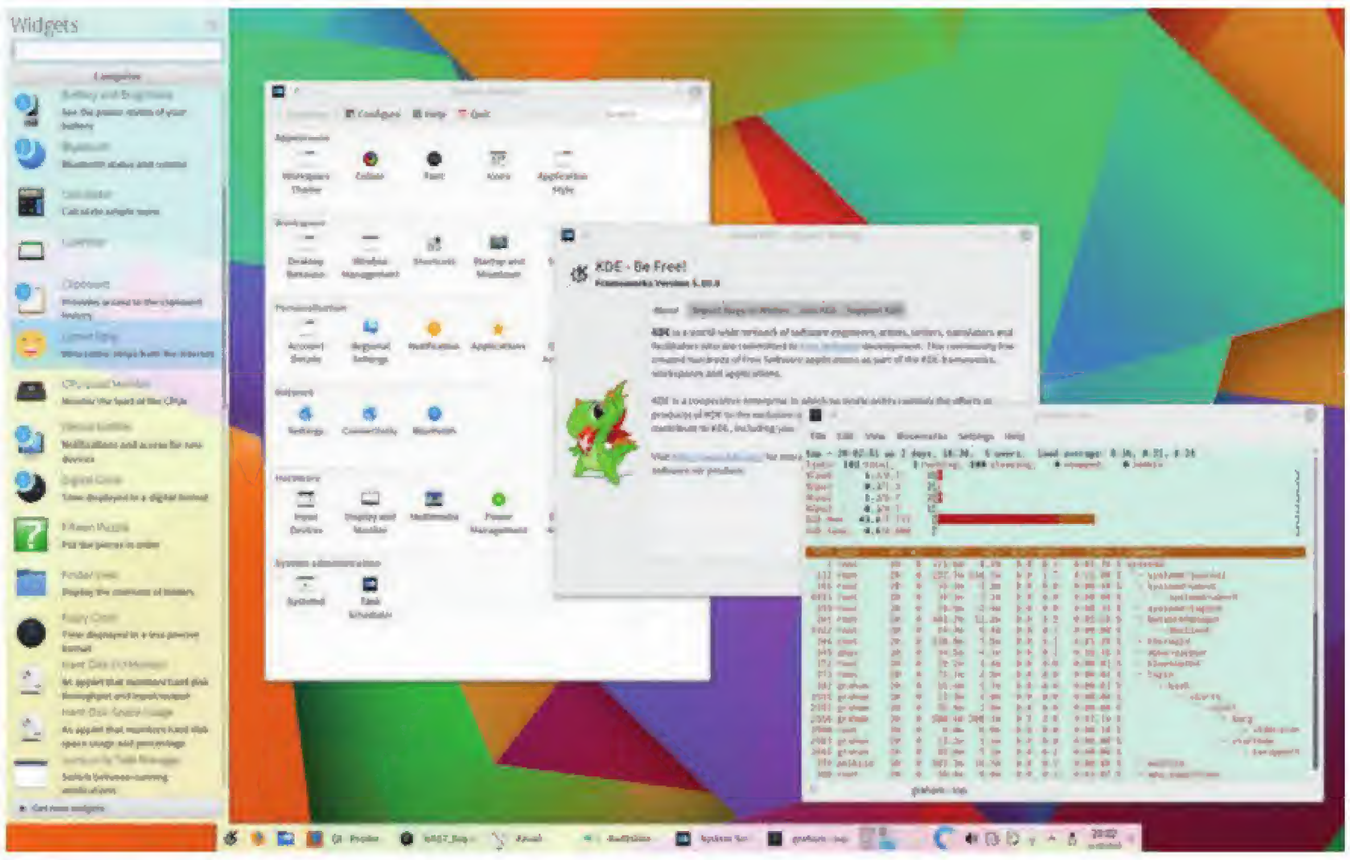
A *Qt* has a long and chequered relationship with Linux. Most importantly, it was the toolkit chosen by Matthias Ettrich when he created the KDE desktop way back in 1996, and it's still a fundamental part of the KDE desktop today. Each major revision of KDE has followed major revisions to *Qt*. *Qt 5.5* should be released in June 2015.

Q Why did KDE choose *Qt* if it wasn't specifically looking for cross-platform compatibility?

A Matthias chose *Qt* because he was looking for a comprehensive graphical toolkit that could generate modern-looking applications. He wanted KDE to be more than a window manager and widget kit, and to do everything from file management to email reading. *Qt*'s libraries made that easier to achieve with a single toolkit.

Q If *Qt* is so good, why didn't Gnome choose it?

A Ooh, this is a good story. *Qt* inadvertently led to the creation of the *GTK* toolkit and even Gnome itself because these were created by a faction of desktop developers who didn't agree with the *Qt* licence. As Eirik Chambe-Eng explained in an interview with **dot.kde.org** back in 2004, "When we started Trolltech we were fascinated by Linux and the idea of free software. At the same time we had neither the expertise nor the finances to do sales and marketing. It was really a very natural and logical thing for us to give away *Qt* for free for free software projects (open source as a term didn't exist back then)."



Qt 5 has taken a lot of features and suggestions from the team behind the KDE desktop.

Unfortunately, this dual licensed open source/proprietary solution was confusing. Not only was the open source version limited to Unix and X11, it was also delivered by its own 'QPL' (Q Public Licence), which failed the Debian Free Software Guidelines and was incompatible with the GNU General Public Licence, despite being compatible with the Free Software Foundation's Free Software Definition. This ambiguity led Miguel de Icaza and Federico Mena to create Gnome in 1997 and to provide an alternative to the rapidly growing KDE.

Q What about the KDE desktop – was that open source?

A Yes, but because KDE is tied so intrinsically to *Qt*, the complete project's licensing wasn't straightforward. *Qt* was becoming very successful as a proprietary product, despite the free version being available. It's currently used by a large number of institutions, including the European Space Agency, Dreamworks, Disney Animation Studios, as well as in lots of cross-platform commercial software like *Spotify*, *Skype*, *Maya* and *Mathematica*. *Qt*'s commercial growth gave Trolltech, the company behind *Qt* at the time, the confidence to re-licence *Qt*. First under the GPL, then with the release of *Qt 4* in June 2005, GPL for Windows. Nokia made the final step to LGPL 2.1 in 2009. *Qt* was even in talks with the FSF when it created GPL v3, and added this as an optional licence. LGPL v3 was finally added with the release of *Qt 5.4* in late 2014.

Q So everything is awesome now?

A We think so. Thanks to *Qt* being a commercial success, it enjoys better quality control and documentation than many open source projects. The commercial and open source versions are identical, but Digia is using the new modular system to experiment with paid-for proprietary modules. These include code optimisers, a purchasing API and a Qt Quick Compiler. This has caused some concern in the community, but no one has yet taken the trouble to rewrite open source versions of those modules. And there was another debacle recently when *Qt* made open source users register before getting access to a download (the project later said this was a mistake). Digia and the *Qt* community knows that open source, KDE and the wider community has been integral to its success, and remains integral to its future, so we're certain it will remain a vital pillar in the success of Linux and open source.

"Science fiction writers shouldn't predict the future in the same way that drug dealers shouldn't take their own product – it never ends well."

CORY DOCTOROW

Ben Everard and **Graham Morrison** meet the author, blogger, journalist, digital rights activist and crazed adversary of all things DRM.

Cory Doctorow once ran a peer-to-peer download service, helped found the Open Rights Group, and is a fellow of the Electronic Frontier Foundation. One of the ideas that kept coming up in our chat is that information doesn't really want to be free, in contrast to the old hacker adage. His books, both fiction and non-fiction, are incredibly popular and challenge the authoritarian direction technology is taking us. And Cory is prepared to live by example. His first book was the first book ever to be released under a Creative Commons licence, and he's committed to how ill suited copyright and DRM are to a modern age never envisaged by early legislators.

We met Cory Doctorow in May after a brilliant talk he gave in Bristol. It was the day of the Queen's Speech, when she opens parliament with a statement of intent written by the election's victors (the centre-right Conservative, or Tory party). The speech was widely rumoured to contain profound reforms to the UK's obligations to the Human Rights Act, reforms that were supposedly cut at the 11th hour, and it was under the shadow of these rumours we started our interview.

LV You've said that future generations will look back at this period as being one in which it was really hard to copy stuff. But with the new Tory government and trade deals going through, do you still have optimism for the liberalising of copyright laws?

Cory Doctorow: I'm very sceptical of optimism. I think that optimism and pessimism are effectively predictions about the future. I think science fiction writers shouldn't try to predict the future in the same way that drug dealers shouldn't take their own product. It never ends well. After all, if you were optimistic about the future, you would get up every morning and do everything you could to make computers better. If you were pessimistic about the future, you would do the same thing. So optimism and pessimism don't really give you a course of action. Hope is, for me, a lot more interesting. Hope is kind of why you tread water if your ship sinks. Not because you have a good chance of being picked up, but because everyone who's ever been picked up has treaded water until rescue came along. And I have hope for the future.

"If you believe you shouldn't start unless you're pure, you won't get anywhere."

I think what the Tories have proposed in this election may subject British people to inordinate computerised risk and will be terrible policy. But I think that the global forces that are making copying easier – which is to say making computers faster, storage more compact and the internet more fault-tolerant – those forces are largely indifferent to whether or not David Cameron expands lawful interception capacity into a realm that can only be dreamt of at Hogwarts.

LV How can the average person help copyright change go in the right direction?

CD: Well there's a bunch of things. Larry Lessig [professor at Harvard Law School] divides the factors that legislate our world into four forces: law, code, norms and markets. And I imagine people who read Linux Voice are positioned to do code, which is something a lot of people are not positioned to do. But there are free software projects that work towards freedom and that need your help. That's things like *TrueCrypt* and *Enigmail*. You know *Enigmail* is supported by one developer – it's a key piece of software and its UI is not great and it needs patches to be brought up to date.

We were just talking about problems with UEFI and full disk encryption. That's another area where people who are coders and administrators can work and contribute and pop out reports and look at open bugs and so on. EFF and the Free Software Foundation both maintain lists of free software projects that are looking for contributors.

There are probably areas in your life where you feel some despair because every month you're sending some money to companies that are working to destroy the world. Maybe the only DSL provider in your neighbourhood is BT, which is an enthusiastic participant in the Great Firewall of Britain. Or maybe it's that you have pre-flavoured mobile devices or laptops, which are harder and harder to get away from. Or maybe your friends require you to use Facebook in order to stay in social touch with them. You'll never be pure. And if you believe that you shouldn't start unless you're pure, then you'll never get anywhere. But one thing you can do is hedge against these compromises you're making. You can

"I think that the future is up for grabs. That we can change the future. That what we do affects the future that arrives..."

help make up for it by tithing some of the money that you spend every month making the world worse, with organisations that help make the world better, whether that's Open Rights Group or Article 19 or Privacy International or the Electronic Frontier Foundation or the Free Software Foundation Europe. All of these groups do amazing work on all the issues that relate to the free and open internet, and they could use your help.

In terms of markets, supporting companies and firms that use free and open source software is a really good way that you can make an impact. It doesn't solve all of the problems, but it solves some of the problems. And then politically, getting involved with political parties and working with them on their information policy is a really important thing. In the last election, I joined the Greens because I think they had a really good information policy. But I know people in the Tories, in the Lib Dems, in Labour and in the SNP who really care about information policy. They come at it from a different angle to me, but they do care. And they are part of the forces that drive their party's policies in one direction or another. So whatever your political stripe, there's probably someone in the party of your choosing who cares about these issues in the way you do and is looking for moral support from within the party, and parties are responsive to their base.

One of the reasons we're not seeing the Human Rights Act repeal being introduced in the Queen's speech is because, within the Tory party, it's a very divisive issue. And so, being part of a noisy voice inside some party is a really good way to make a difference there too.

LV Our magazine is released under CC Attribution Share-Alike licence within nine months. What are your thoughts on that sort of embargo process?

CD: Well it sounds reasonable. I think the real crisis of copyright is that copyright historically has been made as a means of regulating the entertainment industry. So if you look at the history of copyright laws, usually what's happened is that the entertainment industry has had some best practices that it used internally, and they went to parliament or congress and then got it based into law. And the way that we figured out whether copyright law applied to you, whether you were in the entertainment industry, was whether you were making or handling copies. And making and handling copies always implied industrial activity, because books always had printing presses in their history, records always had record pressing plants. But when something copied was non-industrial – everything we do on the internet makes copies – rather than saying, "OK well you know we just need a new way to figure out who copyright applies to", we just say that copyright now applies to everyone!

If you rip a page out of a book and stick it up on the wall, that's covered by copyright law. This is crazy...

"Supporting firms that use free software is a really good way that you can make an impact."

If you're sending an email to your daughter while you're travelling for work, copyright should be the system to regulate that too because, like, why not!

Creative Commons, as useful as it is, is a hack to get around the fact that there's just a bunch of people who are now being bound by copyright who shouldn't be bound by copyright. So if I want to read your magazine, I buy your magazine and read it. I don't have to sign a contract. I don't have to understand a contract. If I want to share this magazine with a friend, I don't have to regulate that behaviour on the basis of a contract. The fact is, if you have to understand the law and then form a legal agreement in order to read a magazine, what's in the agreement is almost irrelevant to whether or not that's good or bad. That situation is bad because it makes no sense, that looking at art or listening to music or reading a book should be a contractually regulated activity.

When your daughter's a few years older [Graham's seven-year-old daughter was assisting at the interview] and she wants to make her first Harry Potter fan website, she will be regulated by the same rules that regulate Warner Bros and Universal, when Warner Bros licenses Harry Potter to Universal to make that Harry Potter theme park. And your daughter will probably not be capable of understanding those rules, not because she's not an exceptionally

Corey reckons the Pirate Party, for all its lack of electoral success in the UK, has played a part in driving the agenda of digital rights within the EU.

intelligent person, but because those laws are made to be passed out by people who have four-year law degrees and then five years of specialist training. And even if she were the Doogie Howser of copyright law and at age 12 was able to comprehend the law and pick up the phone and ring Warner Bros and say, "What are the terms under which I get a licence for my Harry Potter fan website?", no one there would answer her call, because the terms are you can't have one.

But kids have been doing fan-ish activity in relationship to the literature they love for longer than copyright has existed. And the fix to that is not figuring out how to streamline copyright law so kids can understand it, the real fix for this, if we're ever going to make sense of this stuff, is to change who copyright law applies to by redividing the realm of activities into cultural and industrial. There will be things that will be in the middle. Like *50 Shades of Grey* started as fanfic and it was noncommercial, and then it became the most successful book of all time. And those cases are cases where both sides can hire lawyers and ask the judge to decide which law is dispositive when. But in almost every other case, where we find ourselves asking, "Is this the right Creative Commons licence?", "Is this the wrong Creative Commons licence?", the reality is that if we were dealing with physical goods, nobody would be asking about licences anyway. And any answer to that question that doesn't start with that is giving too much credit to sense that is really a nonsense.

LV If copyright is an obfuscation and we're always going to be chasing that, and Creative Commons is a hack, what should we do?

CD: I think we can define a suit of activities that we think of as industrial, whose litmus test isn't 'are you copying?', but whose litmus test involves something more complicated like commercial activity and activity that is not cultural, so it should be both commercial and not cultural.

At that intersection we can define statutorily or we can define them in principle, or we can define them with a set of examples and then a set of principles for evaluating new activities that are made possible by technology. If you are doing something that is both commercial and not cultural, you should be bound by copyright law. Otherwise, if you're bound by any rules, they should be cultural rules. Plagiarism isn't always a copyright violation. If I claim to have written Shakespeare and put my name on the cover of *The Tempest*, I've committed no copyright law violation. Normatively, culturally, I've done something really bad. They might be formal rules, they may be normative rules, but they won't be copyright's rules. Which rules does copyright have? Well that matters a lot to me as somebody in the industry supply chain, but almost everyone else in the world

can just ignore copyright. We should start by saying, these activities are commercial, these activities are cultural; if you're not doing something commercial and you're doing something cultural, then you're governed by a different set of laws.

Want to read more Doctorow? There are plenty of books of his available on all good internets.

LV Is there something we could create to run in parallel to Creative Commons?

CD: No, unfortunately there isn't. This is an area where we need legislative reform because the copyright law works under international law... You automatically get a copyright for your life plus 50 years.

Maybe you could get some students to say that some classes of works aren't in copyright, but I mean effectively we have to break the legal deadlock in order to make sense of this. That doesn't mean we should give up on Creative Commons. I believe in Creative Commons licences, support them financially, I use their licences, I promote their use because it's the next best thing to fixing this untenable hairball of legal gubbins. To pretend that the problem isn't an untenable hairball, is to ignore the problem. We can only fix so much around the edges.

LV Why do you think licensing has really caught on in the tech world of open source software, but has been slow in other areas like publishing and music?

CD: Code is, on some fundamental level, maths. And maths is science. And science has, since the enlightenment, operated on the basis that everyone else has to be able to replicate what you do and build on it. So, if you look at the history of open licensing, it actually came about really when a commercial entity started to assert copyright on paper tapes. It started when Richard Stallman walked into the lab one day and the paper tape drawer was locked, and no one would give him a key to the drawer with the paper tapes in it.

It was a collision between an enlightenment ethos, that says that you have to tell other people what you know in order for knowledge to be collectively advanced, and this industrial drive to property-ise information.

Think of it from the perspective of open eHealth records. I live near Moorfields Eye Hospital, which is the best eye hospital in the country, possibly in the EU, and after the NHS eHealth record system collapsed, they hired an open source developer named Chris Reading to build a LAMP stack glaucoma tracker called *OpenEyes* (see **www.openeyes.org.uk**). And I went and gave a talk for them to other eye surgeons about why this makes sense and why you should use this open version instead of paying one of the big consultancy firms to build you a proprietary glaucoma tracker. And I said, leaving aside all the commercial considerations, when you put a wing on the side of your hospital and the firm of engineers comes in and says we're not going to tell you how to calculate the load stresses on this RSJ (rolled steel joist) because that's proprietary, we're not going to tell you where the trunking is in the walls, because we want to make sure that you pay us when you want a new mains outlet.

You would say, commercial considerations aside, that this is not about whether you deserve to make money, it is just not right. It is not responsible for us to have a hospital where we can't independently verify the way that you calculated our load stresses.

And so, when the consultants come in and says our software engineers aren't going to tell you where the software RSJs are and where the software trunking is in the walls, they are effectively saying our business model trumps best engineering practice, and for that reason alone, leaving aside all of the questions about efficiency and code quality and cost, they should never ever be buying proprietary software.

Firms have dealt their commercial advantage out on the basis of their ability to deliver to deadline, their ability to liaise with other firms and gather their requirements and reflect them back to them in their projects that they build and so on, but not on secrets about how they accomplish their stuff. The secrecy is in the culture of the firm, that's their proprietary secret source, not in the standard tools that they use. No one uses a secret bulldozer, right? And no one uses a secret RSJ. And nobody should use a secret operating system for anything that matters.

LV For your books, you use a CC non-commercial licence.

CD: Non-commercial, and then some of them are Share-alike and some are non-derivatives.

LV Does much come out of the Share-alike books?

CD: Yeah, there's a fair bit. Mostly, the only 'problem' I have is the order in which the translations came out. So what I found is that foreign publishers were by and large OK with translations,

provided that it didn't surprise them. Provided that I sat down and talked to them and said 'do you know when you buy this book and do a translation, fans are going to be able to do their own translations too'. There will be a varying quality, but their primary motivation is not to compete with you, but to improve access, to demonstrate and improve their own mastery of the language, to do something with a group of friends.

My foreign publishers were totally OK with this, but they went into a meeting at the Bologna book fair with my foreign rights agents who pitched them on a book, and they said "but there's already this translation out there." They could never make sense of how that translation could operate alongside their commercial edition.

LV Were you the first publisher to use the Creative Commons licence?

CD: My book was the first ever Creative Commons book (*Down and Out in the Magic Kingdom*). It came out the same week as the Creative Commons licences.

"Nobody should use a secret operating system for anything that matters."

LV Was that a difficult conversation to have with your publisher?

CD: Back then my publisher was like 'we're paying you $7,500 for your first novel, we're printing 10,000 hard covers, we break even if we sell 4,000 of them, what's the worst that could happen here?' And a lot of people said 'Doctorow can afford to do this, no one's ever heard of him, what does he have to lose?' Now I hear a lot of, 'well Doctorow can afford to do this, he's so well known, he can afford to do it'.

You're kind of damned if you do, damned if you don't. I don't think Creative Commons licences make people care about your books. But what they do is they make it so that people who do care about your books find it easier to share them, to promote them. That's been the guiding light of my CC philosophy. My editor at Tor Books, Patrick Nielsen Hayden, is super Linux savy. Like, I met him on a BBS in the 80s, he administers his own Linux boxes. I went over to his place once and I was like "What browser is this!?", and he said "Oh, it's *Konqueror*". That was the first time I saw *Konqueror*! So he's pretty tech savy – he gets this stuff. He was writing Google type plugins back when I was doing this stuff. So it was a very easy sell.

LV Have you had to walk away from any deals because a publisher wouldn't accept it?

CD: Kind of. I have a picture book coming out next year from another division of Macmillan, which is the same publisher that does my novels, Tor. And the picture book had originally been sold to a giant multinational company. I'm not going to name them, but they're the largest publisher in the world. The guy who bought it was this brilliant guy who was their head of digital strategy. And he knew where I stood on DRM and I knew where he stood, and we were all cool with it. We worked on it for years, on this book, and went through lots of revisions. It wasn't all that we were doing, it was just every now and again that he'd send me revisions and I'd get to them and I'd send them back, or we'd get sketches from an artist, and then go to a different artist. Finally we're ready for the contract... And months went by and my agent was bugging him and finally he sent me an email from his non-work email address and said could you call me on my non-work phone. And he said, you know, I've been over it with contracts but they won't do a contract without DRM, and I've tried everything. I said look what if we just don't buy the eBooks? And they said no, we have to now. And he said 'OK, what if we use the covenant not to use the eBook rights unless we have mutual agreement?' And they said no we can't do that either. And I said look, here's the pro forma spreadsheet, nobody buys picture books in electronic form. Our pro forma earnings after we pay for the conversion is minus £80 on this eBook, we've already sunk thousands into this eBook, are you crazy? And they said no, we can't do it.

And that's when he said, "so that's when I quit my job". So he quit his job that day to go to work for another publishing start-up that was doing amazing digital publishing stuff. Macmillan snapped up the book and now it's coming out in 2016. It's called *Posy the Monster Slayer* and it's going to be a fun book. It's about my daughter, who is your age (talking to Graham's daughter), who one night, after she's been given all these super girlie toys for her birthday like a Barbie® Dreamhouse™ and a tiara and all this junk, and she eats too much ice cream and cake and when she goes to bed, she has nightmares that the monsters are coming. And so she takes all of her girlie toys and she turns them into weapons and she kills all of the monsters with them. It's pretty fun! So anyway, all's well that end's well. LV

Next from Doctorow: a parable on the dangers of eating too much ice cream.

LINUX VOICE REVIEWS

The latest software and hardware for your Linux box, reviewed and rated by the most experienced writers in the business

Andrew Gregory
is hayfever-free after moving so far north that nothing can grow.

John Deere is a company that makes tractors. Those tractors have engine management systems, and the engine management systems need computer code in order that they may manage the engine. So far, so innocuous.

However, in the wonderful world of proprietary software, John Deere claims that, because of the presence of this code, when you hand over your money to buy your dream tractor, you don't actually own it – you're merely licensing it. Despite the tractor being physically present in your field, it isn't yours; it still belongs to the company.

Property rights

This is really, really weird, as the EFF has been telling us since this story broke in April. But what's been overlooked is the question: in what way would John Deere lose out if it open sourced the software in question? I can't imagine horny handed sons of toil sitting around a laptop in the barn debating the merits of Python 3 vs 2 in the implementation of this year's sowing patterns. But I can imagine a scenario in which the increased serviceability of its tractors made John Deere more popular among folk with an eye for a bargain, as farmers usually are. Silly buggers.

andrew@linuxvoice.com

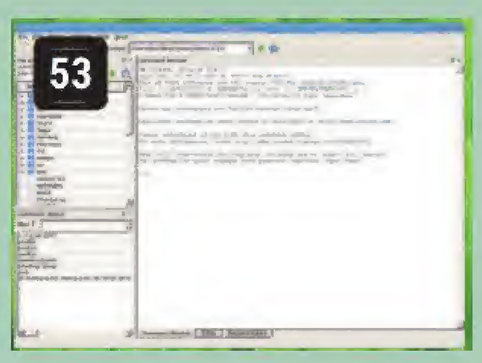
On test this issue...

Cinnamon 2.6

Mike Saunders explores the desktop to end all desktops – it works as you expect it to, it looks great, and it won't cause Gnome vs KDE arguments.

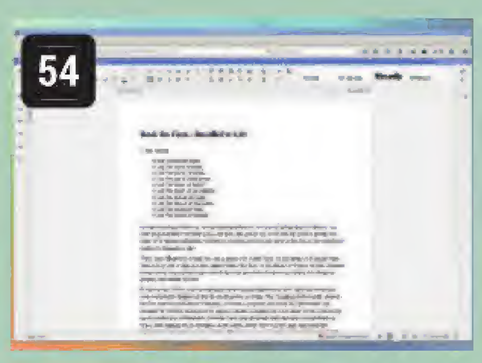
Fedora 22

Committed KDE fan **Graham Morrison** steps out of his comfort zone to try a Linux distro with Gnome's footprints all over it – and absolutely loves it.



Gnu Octave 4.0

Ben Everard likes graphs, code, and playing with huge data sets. This programming language is right up his street.



OnlyOffice

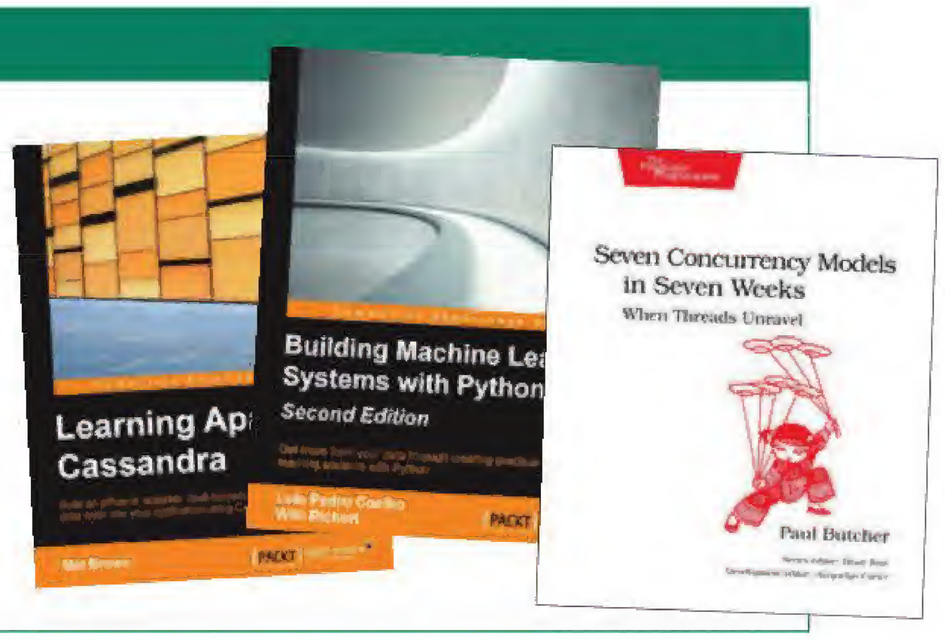
Ben Everard also likes the convenience of a cloud-based office suite – especially when he's not being spied on.

Scribus 1.5

What is still quaintly called desktop publishing has a Free Software champion – **Graham Morrison** tests the new features.

BOOKS AND GROUP TEST

The Raspberry Pi has outgrown its makers' intentions several times, and the latest incarnation – Version 2 – now has enough grunt to function as a usable desktop computer, running a web browser, productivity software and more, all for a bargain price and minimal power consumption. Our challenge this month is to find the best distro to take advantage of this new power, whether that's an old favourite optimised for ARM, or a completely new creation. And in the book review pages, we learn that the printed page is still a viable medium for learning!



Cinnamon 2.6

Linux Mint's Gnome 3 fork has come a long way in the last four years. **Mike Saunders** explores this shiny new release.

DATA

Web
http://cinnamon.linuxmint.com
Developer
Linux Mint and others
Licence
GPL

Forks in the free software world are often regrettable events, and can result from nothing more than personal squabbles between developers, but Cinnamon demonstrates that forks can be productive as well. Back in April 2013, with the release of the Gnome 3 desktop and its radically redesigned Gnome Shell, the Linux Mint distro team was left with a dilemma. On the one hand, the Mint developers wanted their distro to stay fresh with the latest desktop technology – but on the other hand, they were concerned about the impact of Gnome 3's redesign. After all, Mint prided itself on offering a slick all-round desktop OS with a familiar user interface, whereas Gnome 3 looked drastically different and was crafted with tablet and touch interface users in mind.

"Cinnamon 2.6's user-facing improvements, performance tweaks and buxfixes all add up."

So the Mint team took the difficult step of forking Gnome. This was received with mixed reactions by the wider Linux community: do we really need more forks? Will it just die out when the Mint crew run out of energy? Why don't they just use Mate (a Gnome 2 fork) instead? Well, here we are over four years later, and Cinnamon has gone from strength to strength. Whereas early releases of the desktop were basically the Gnome 3 codebase with some design tweaks to make it look like Gnome 2.x releases, with Cinnamon 2.6 many individual Gnome programs have been forked and the two codebases are now distinct.

Cinnamon is still very much associated with the Linux Mint distribution, but the desktop is available in other distros such as Fedora. We tried Cinnamon 2.6 by installing Mint 17.1, the latest available release at the time of writing, which included Cinnamon 2.4. We then added the Romeo repositories for packages that are still undergoing testing – and one **apt-get update && apt-get upgrade** later (and 500MB of downloads) we had the sparkling new release to try out.

So what's new?

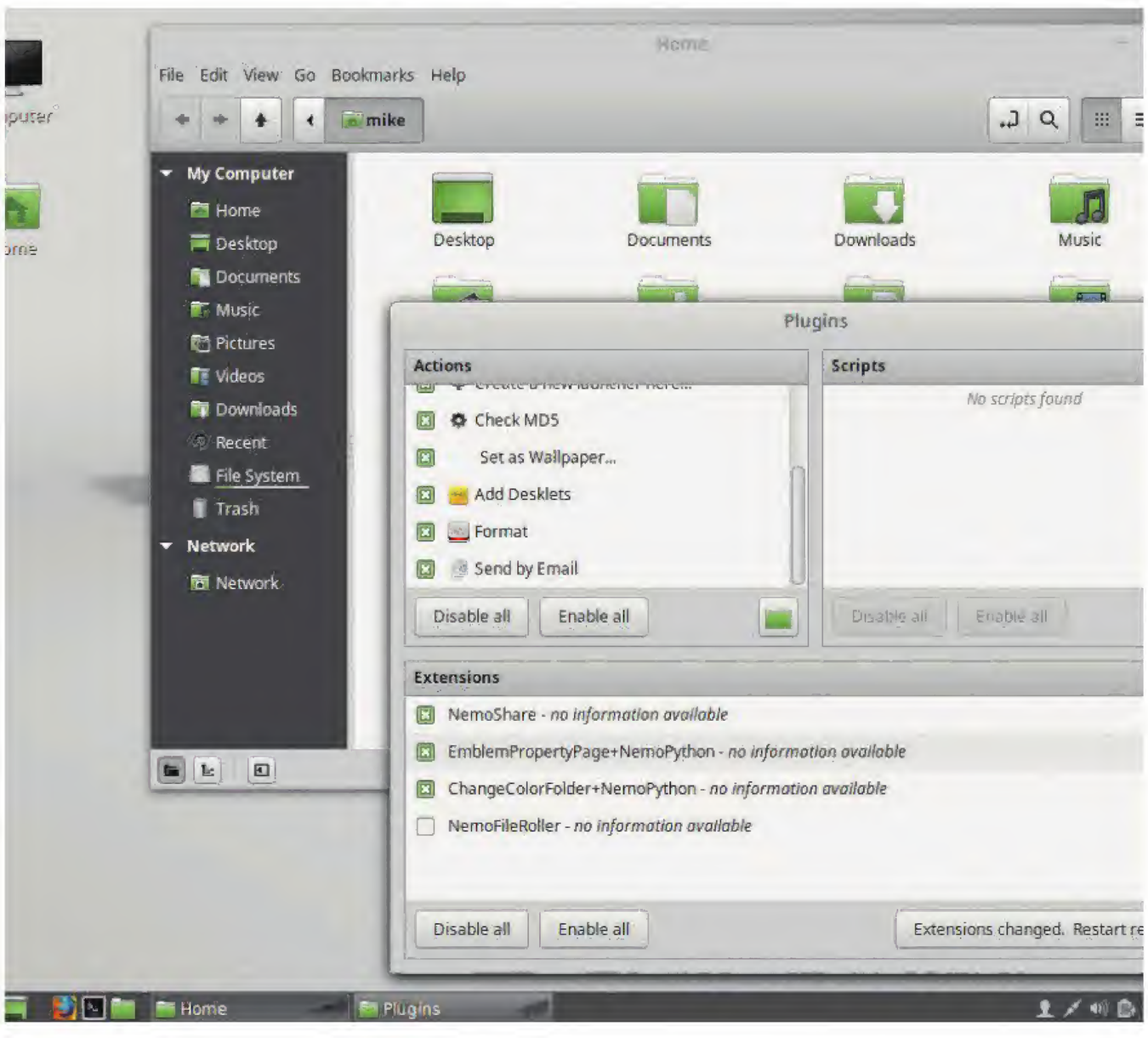
Desktop environment upgrades can often inflict a noticeable hit on performance, but we found no change between Cinnamon 2.4 and 2.6. Both started up in the same time, and the stock Linux Mint installation, freshly booted up, occupied around 330MB of the RAM banks with both versions. So Cinnamon will run decently in 1GB of RAM, but it's far from being a lightweight desktop in the realms of *Xfce* and *LXQt*. Mint's Cinnamon themeing is excellent; the desktop looks polished and tidy, with subtle effects and few distractions.

The main menu organises applications into categories, and also features shortcuts to commonly used programs down the left-hand side. Meanwhile, the panel along the bottom is very much Windows-esque, with a taskbar, system tray and clock. Some Linux users might deride this setup as being rather unimaginative, especially with Gnome 3 and Ubuntu's Unity adopting more novel approaches to window and desktop management, but familiarity (and ease of transition from Windows) was always one of Mint and Cinnamon's goals.

We've never had serious stability problems with Cinnamon over the years, but the developers have added a new shortcut for this release: Ctrl+Alt+Esc. This restarts the *Nemo* file manager and *cinnamon-settings-daemon* process, effectively restarting the desktop environment as a whole, and is provided in case of a freeze (which has been reported in earlier releases, according to the developers). This key combo doesn't restart applications, however – so if you're doing something in *LibreOffice* and Cinnamon locks up, you can restart the desktop without having to log out and lose all your work.

In terms of user-facing improvements, the System Settings dialog has been redesigned to be simpler and more attractive. Options are neatly divided into sections accessible via buttons along the top, with silky transitions between them. New window

This release doesn't look drastically different to 2.4, but there are many small and subtle changes all around the interface.



animation effects have been added as well, with a great deal of customisation available for how quickly they perform. On the desktop, the power applet has been rewritten to consolidate multiple features in the same place: along with battery information, it also provides controls for screen brightness (and if applicable, the keyboard backlight).

With Cinnamon 2.6, it's now possible to have multiple panels operating independently, and a new applet called *Inhibit* has been introduced which prevents notifications and power management from interrupting presentations and similar work. The sound applet has been updated too, with better *PulseAudio* integration and the ability to change sound levels for individual applications.

Nemo, Cinnamon's file manager, sports a new plugin manager for single-click enabling actions, scripts and extensions that can be applied to files and directories via the right-click context menu. This context menu has also been cleaned up to show only the most common operations, while large file operations are now queued up and performed sequentially, rather than in parallel like in previous releases.

In previous versions of the desktop, the *cinnamon-screensaver* tool did little more than lock the screen after a certain amount of time; with Cinnamon 2.6, screensaver support is much more extensive. You can use fancy *XScreenSaver* modules along with HTML 5 screensavers, and customise many more settings. Other user-facing changes include improvements to the accessibility tools, including the magnifier, mouse zoom modifier and on-screen keyboard (which now has an auto-hide facility).

Finally, a proper screensaver! Now Cinnamon can use fancy screen locking effects from *XScreenSaver*.

Beneath the surface

Under the hood, a significant change has been implemented to improve the desktop's portability. Cinnamon can use both *ConsoleKit* and *Logind* (the latter is from *Systemd*) to handle user logins, but in previous releases this had to be defined at compile time. With version 2.6, you can choose which framework to use by changing a setting.

Meanwhile, the Cinnamon team has done a lot of work to reduce CPU usage and avoid duplicated actions. The main menu is now drawn six times less frequently than before, while optimisations in the window manager have reduced its overall CPU usage by 40%. To speed up the desktop's load time, Cinnamon 2.6 now has a preload feature that caches themes and application information earlier in the boot sequence, so after login the desktop appears much more quickly. Optimisation work is difficult and tedious, and most developers would rather be working on flashy user-wooing features, so we give a big thumbs-up to the Cinnamon crew for their efforts in this area.

For application developers, Cinnamon 2.6 is the first release to include proper documentation (see **http://developer.linuxmint.com/reference**). It's a mixed bag and has some major holes, but at least it's a start and provides some useful tutorials for creating applets for the desktop. Because multiple panels are now supported, many applets will need to be updated, especially if they assume they will only ever be installed on a single panel.

So, is Cinnamon 2.6 worth the upgrade from 2.4? This is a bigger question than it sounds, because for most users it will involve more than just grabbing a few packages; it will mean a major distro upgrade. Linux Mint 17.2 ("Rafaela") is due to be released around the end of June, so it should be available to download by the time you read this. It's well worth the upgrade, even if you have to grab other packages in the 17.2 update as well. The user-facing improvements, performance tweaks and bugfixes add up, and with the new documentation it's turning into a serious competitor to the established desktops.

Cinnamon vs Mate

Both Cinnamon and Mate were born out of dissatisfaction with Gnome 3's redesign, but the two desktops had very different goals early on. Whereas Mate was happy to continue with the Gnome 2.x codebase despite its age, the Cinnamon team was eager to use code and components from Gnome 3 – just with a more familiar design. Mate is still going strong today and the codebase is receiving many updates, so in some respects it's getting closer to Cinnamon. There's also now an official Ubuntu flavour that bundles Mate as the default desktop. But could the two ever merge? One can dream...

LINUX VOICE VERDICT

Everything a desktop should be: attractive, fast, familiar but still customisable. KDE and Gnome have some real competition now.

★★★★½

Fedora 22 Workstation

Graham Morrison tests a distro that uses the Gnome desktop – and likes it!

DATA

Web
https://getfedora.org
Developer
Fedora/Red Hat
Licence
Free plus proprietary firmware

Fedora 22 was released at the very end of May, but we've been using the betas in earnest since our monster distro roundup last issue. As is often the case with Fedora, this release is mostly a revision rather than a revolution, with a few modest refinements and the latest packages. Those refinements this time are unlikely to have a huge impact on your Fedora experience, but they're also quite substantial in the way they change Fedora's internal plumbing. That we've experienced no major issues, even with the betas, means these refinements can only have happened after considerable effort and planning.

The most significant of these refinements is of course the new package manager. If you think about how intrinsic package management is to the fabric of your system, it's a significant success that the switch from *Yum* to *DNF* (*Dandified Yum*) barely flutters a neuron. *DNF* is almost entirely equivalent to *Yum* on the command line, accepting nearly all the same arguments. Of the few that are missing, **--skip-broken** is the guilty favourite we're going to miss most. Even when muscle memory types **yum** by mistake, you'll find **yum** is now a simple script that politely reminds you of its own imminent demise before passing on your arguments to *DNF*.

"It's a significant success when the switch from Yum to DNF barely flutters a neuron."

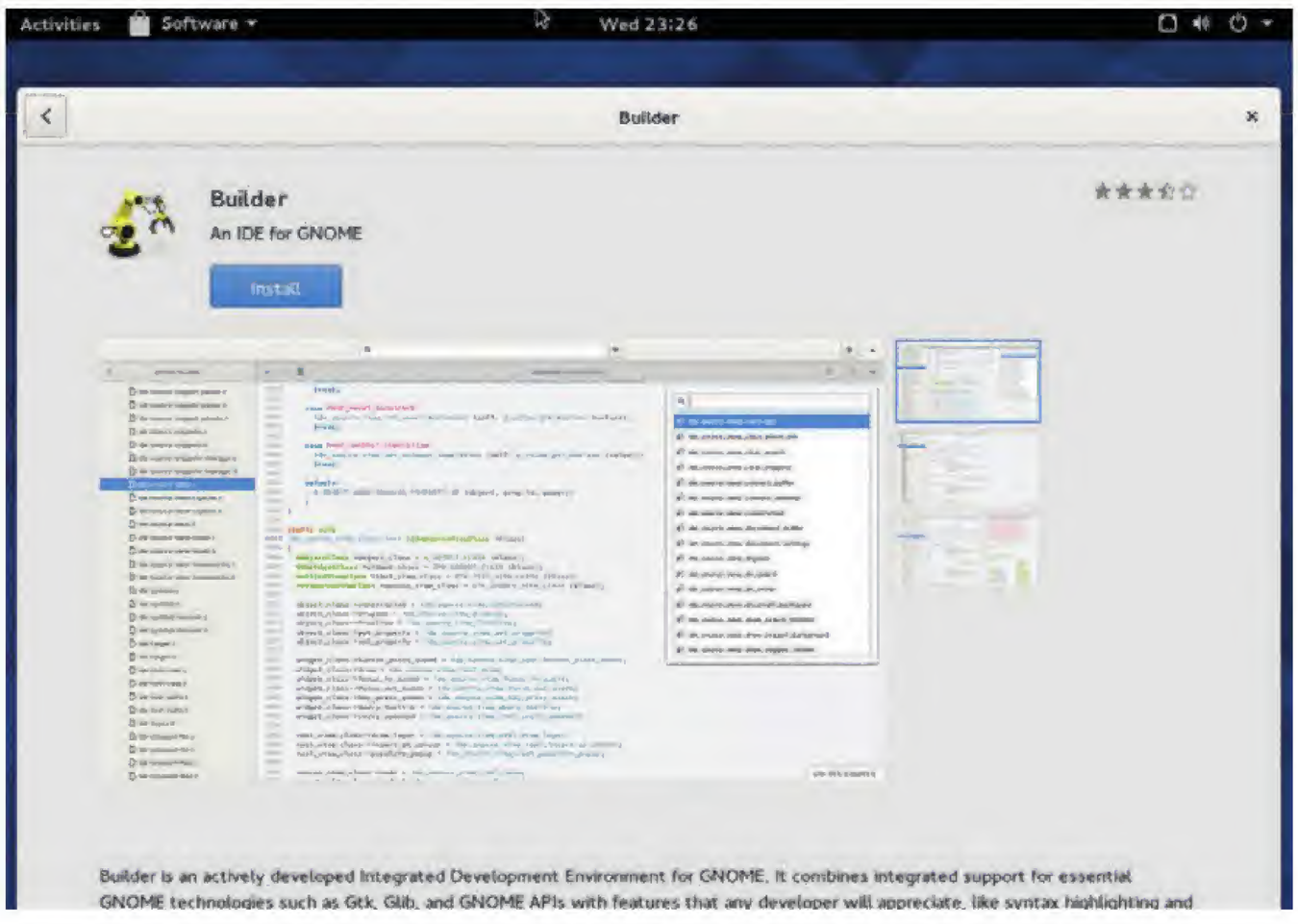
Behind *DNF* is Hawkey, the new API for packaging that's responsible for resolving dependencies. The packages themselves are delta RPMS. Deltas contain only the difference between the installed and the new versions of a package, and take considerable load off your network. But they do require some local processing to create an installable package. From our perspective in rural Hobbiton, we approve of this change, but it would be useful if the package manager recognised from your network and CPU combination that installation may be quicker from a full RPM download rather than a reassembled RPM and offered you the option to revert to full-fat RPMs.

Fedora is now available as Workstation, Server and Cloud spins. Workstation is designed for desktop users.

Gnome Home

The other major feature that most users are going to notice is the inclusion of the latest Gnome desktop, version 3.16 (see our review issue 15). Gnome isn't specific to Fedora, but Fedora always does a good job at creating a default environment, and we're really starting to enjoy Gnome. We love the new grey look and the new notifications. These appear from the middle of the top of the screen and you can now interact with some, such as the calendar or an incoming message.

It was reported that Python 3 would become default in Fedora 22; that transition has been pushed back to Fedora 23. We're used to dealing with this problem in Arch, where version 3 is already the default, and it does create considerable difficulty for Python users. Finally, the other major upgrades include *GCC 5*, which is now the default compiler, and the inclusion of KDE Plasma 5 - its most significant endorsement. We were also impressed by the inclusion of the shiny new Gnome Builder IDE. It's in alpha but well worth a look (see p104 for our getting started guide). LV



There's lots of cutting-edge software in Fedora, including the very latest release of Gnome's new development environment.

LINUX VOICE VERDICT

A strong release by virtue of good upgrades and stability despite major changes to its infrastructure.

★★★★½

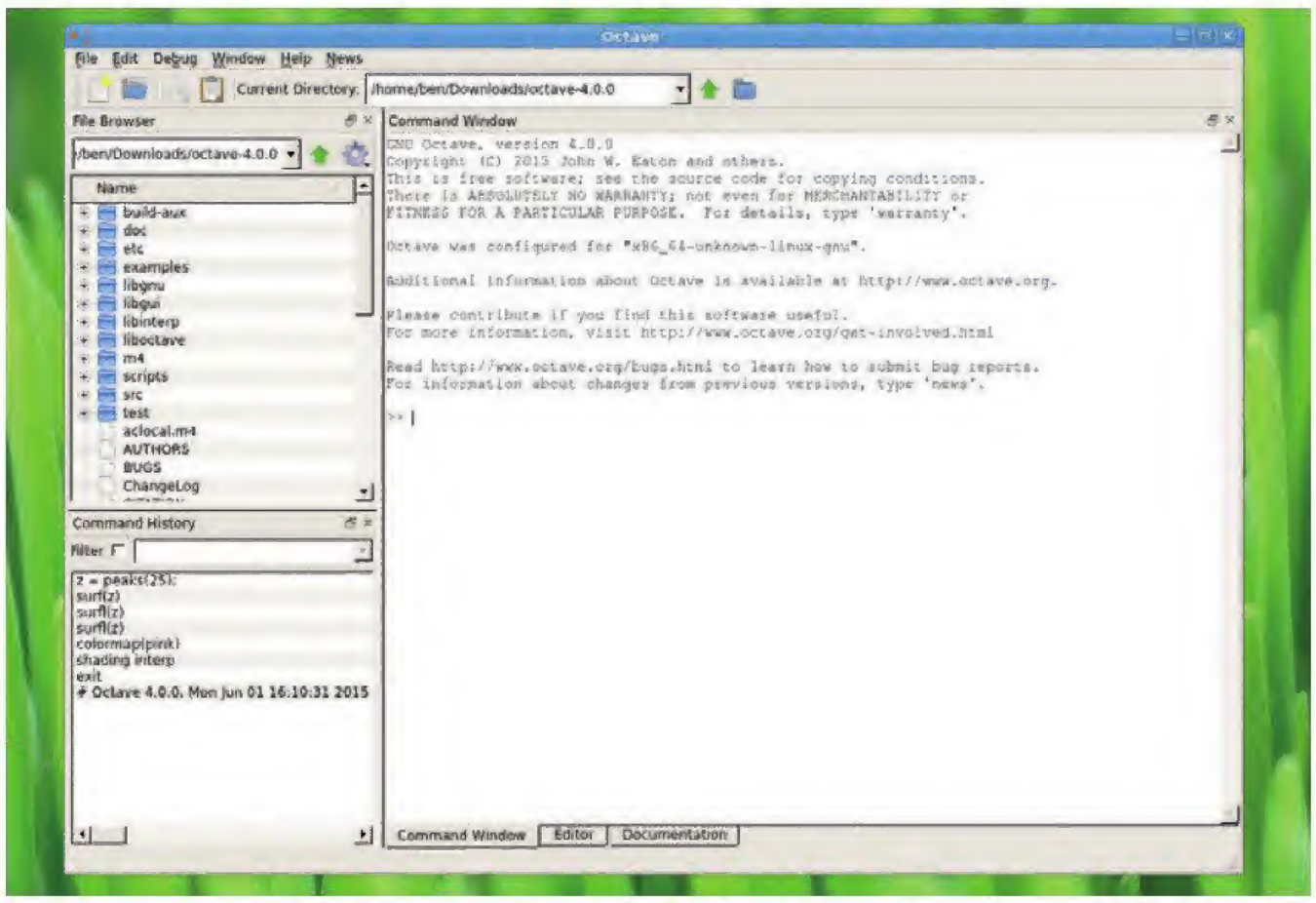
Gnu Octave 4.0

Messing about with mathematics might not be everyone's cup of tea, but **Ben Everard** finds a way to make it more fun for everyone.

Gnu Octave is a mathematical programming language that's designed to help users analyse and visualise numerical problems. It first came out in 1988 and gets a major release on average once every eight years. With such a conservative development pace, 4.0 is a big release, and it comes with a killer new feature: a graphical interface.

Prior to this version, Octave ran in a terminal window. Octave 4.0 comes with a *Qt* graphical user interface, which uses OpenGL to render graphics. Since Octave is often used for visualising data, this should help it take better advantage of graphics hardware when rendering large data sets.

The GUI doesn't add anything new to the functionality of Octave; it just makes it a little nicer to use. Most of the window is taken up with the text interface to Octave that's exactly the same as it running in a terminal. This command window (which isn't an individual window, but a pane in the main window) also has tabs for a command editor and documentation. Additionally there are panes to show files, workspaces and the command history. All these are moveable to other parts of the window.

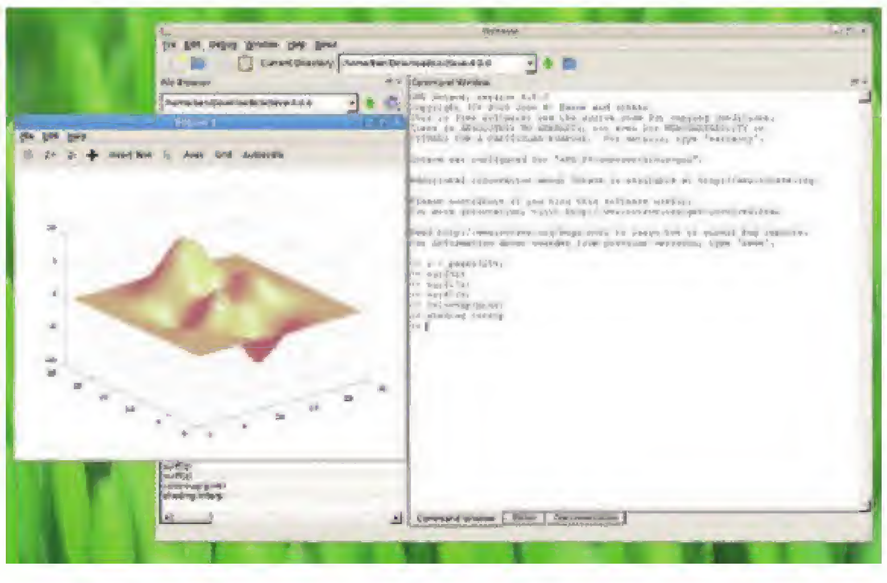


The new GUI has a clean, uncluttered interface which is easy to use even for people unfamiliar with Octave.

Graphical gloss

New people coming to the software might find the different interactions in the command window and the editor a little confusing. In the command window, pressing Tab completes a command, while in the editor, a drop-down list automatically appears giving options for command completion. This is obviously because of the different heritage of the two forms of input, but the difference seems confusing.

Console fans haven't been abandoned, and you can start the software in terminal mode by starting Octave with the **--no-gui** flag, or **--no-gui-libs** if you also want to use terminal-based plotting. You can also change the plotting library from the default *Qt* to either *Fltk* or *Gnuplot* using the **graphics_toolkit()** function.



Visualisations have always been a speciality of Octave, and in version 4.0 they're rendered in OpenGL.

Perhaps the biggest draw bringing people to Octave is its compatibility with Matlab. Matlab is another high-level mathematical programming language that's long been popular at universities. It does have a Linux version, but it's proprietary and licences can be expensive. Octave isn't perfectly compatible with Matlab, but it's close. Version 4 brings even better compatibility in quite a few areas. We won't go into them all in detail, but all the improvements are listed in the release notes at **www.gnu.org/software/octave/NEWS-4.0.html**.

Many pieces of Matlab code will work out of the box, and it's usually not too hard to port those that won't. This makes Octave a great choice for people who have been trained in Matlab, but want to switch because of either financial or ethical concerns over the proprietary model.

The graphical interface of Octave version 4 is a major step forward, especially considering that all the alternatives (such as Matlab and iPython) have great, well tested interfaces. Even though the core of the language works well, without a GUI, it's hard to see how Octave could have stayed relevant in the face of some excellent competition. The GUI lifts the software from an obscure piece of command line software used by a few geeks to a genuine option for teaching and investigating mathematical phenomena.

DATA

Web
www.octave.org
Developer
John W Eaton *et al*
Price
Free under GPLv3

LINUX VOICE VERDICT

The new graphical user interface makes Octave more accessible to new users.

★★★★½

OnlyOffice

Ben Everard reclaims his privacy with a web-based office suite he controls.

DATA

Web
www.onlyoffice.com
Developer
Ascensio System SIA
Licence
AGPL

OnlyOffice is a web-based office suite similar to Google Docs or Microsoft's Office 365. However, unlike its competitors, OnlyOffice is open source (under AGPL) so you can run it on your own server (there's a hosted version available as well).

OnlyOffice is a rebranded version of TeamLab Office, which has been around in one form or another since 2009, so it's had time to mature to a featureful, stable platform. However, it was closed source and Windows-only until the end of 2014, so is still fairly unknown in the Linux world.

There are two parts to OnlyOffice: the community server and the document server. The community server is for collaboration and includes document sharing, and other tools for working as a team. The document server is just for viewing and editing documents. OnlyOffice also releases a mailserver, but this isn't developed in-house, instead it's a build of open source mail tools including *iRedMail* and *SpamAssassin*.

"The real star of OnlyOffice is the online office suite."

You can install OnlyOffice on top of a distribution just like you would any other software, however, there's also a series of Docker images, which means you can deploy the software with a single command. The images are in the Docker hub, but the Dockerfiles are on GitHub so you can modify them to customise the build (**https://github.com/ONLYOFFICE**).

The OnlyOffice Community Server is most useful as a web-based document sharing tool. In this role, it works well, but doesn't offer much to distinguish it over the competition (such as Seafile or OwnCloud). In addition to the document sharing, there's also an online email client. This connects to an email server that could be the official OnlyOffice build of *iRedMail*, or could be any other mail server that supports the usual protocols. Again, this is perfectly functional, though unremarkable, and probably not enough to convince anyone to use a new collaboration tool.

There are some theming options available for anyone who doesn't like the default OnlyOffice colour scheme.

Word processing power

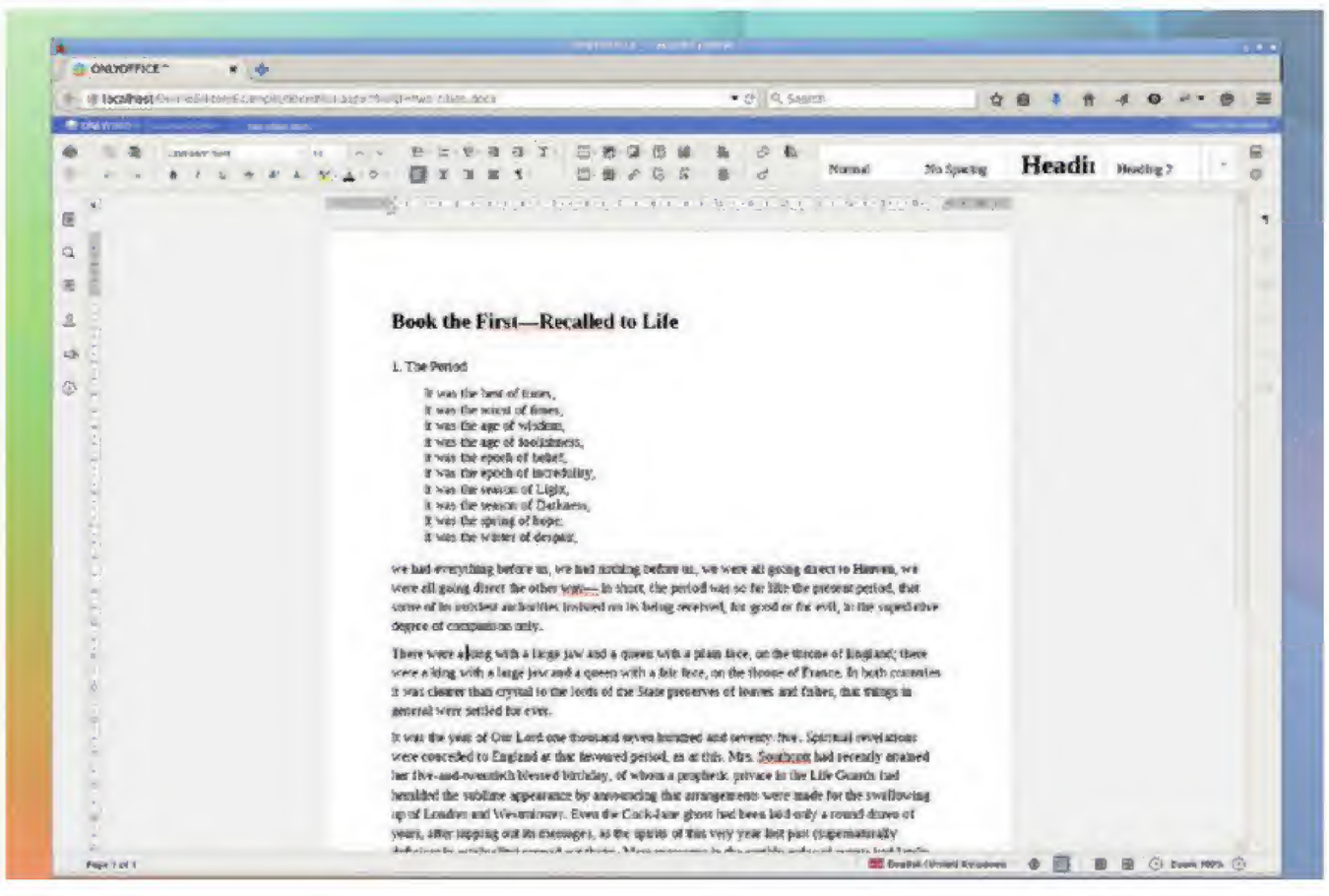
There's also a calendar and an online chat tool. The Community Server can link to any online storage that offers WebDav access (such as Box or OwnCloud). This is a great option if you want to take advantage of OnlyOffice Documents while still using your existing cloud storage option.

The real star of OnlyOffice is the online office suite – the word processing is the best open source web-based document editing experience available. This may change when *LibreOffice* online is released, but for now, nothing else we've seen comes close in terms of experience. It can handle complex layouts, it has plenty of features and it runs well (provided you have a modern web browser). There's also a document viewer that can be embedded in other web pages to allow you to share read-only access with the world.

Our biggest complaint is that OnlyOffice Documents uses Microsoft's DOCX, XSLS and PPTX formats. It can handle others (such as ODT), but only by first converting them into DOCX (they can be converted back before downloading). This probably makes sense from a pragmatic point of view, but it's disappointing from a document freedom perspective.

The spreadsheet and presentation editor are similarly impressive. While none of the office suite has quite the range of features you'd expect of a native suite, they have enough capability for most tasks and far more than other web-based office suites.

The word processor uses a HTML 5 canvas element to display the editor, so you'll need a modern browser – we didn't find any problems in the common Linux options.



LINUX VOICE VERDICT

The best online office suite backed up by a reasonable collaboration server.

★★★★★

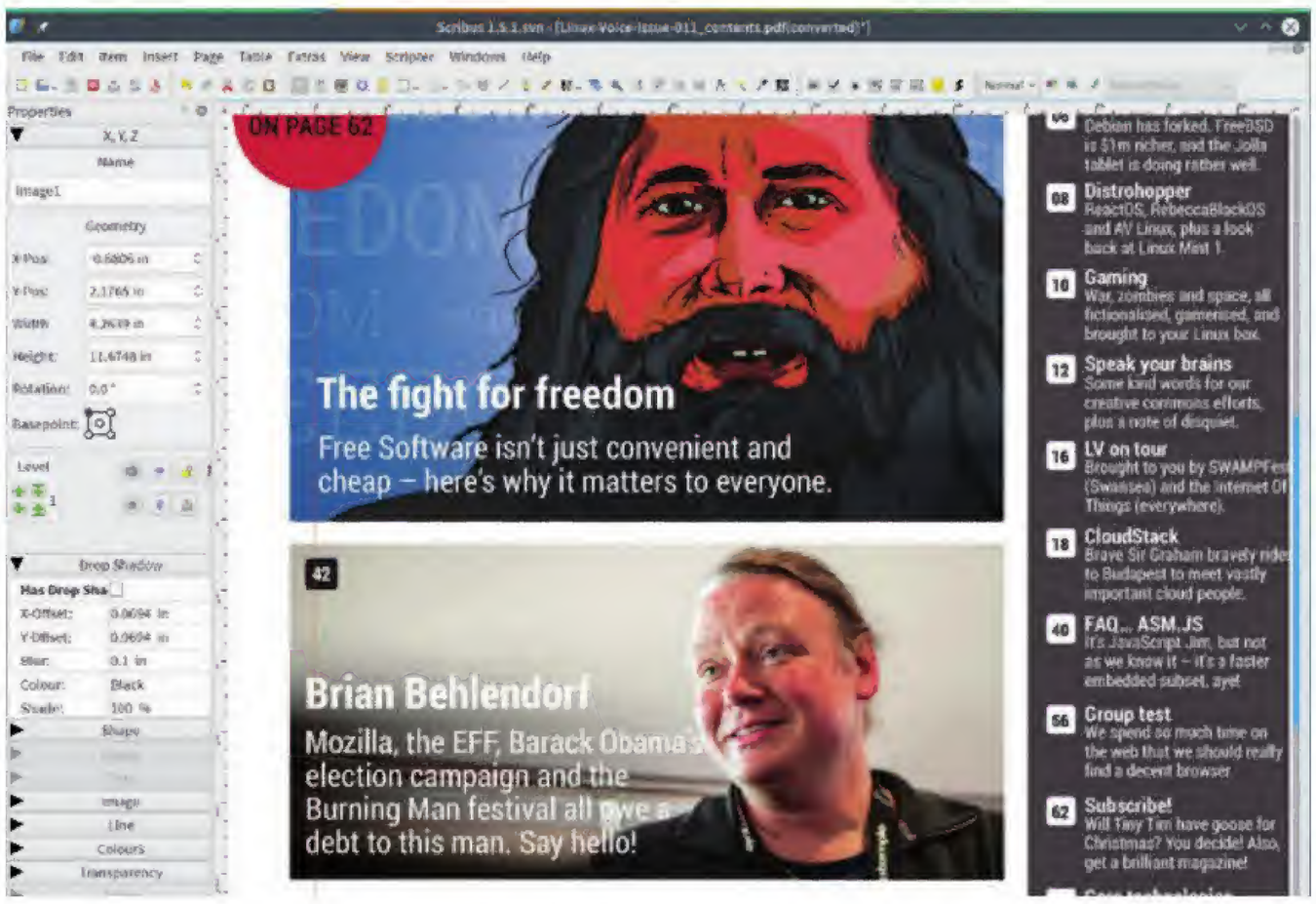
Scribus 1.5.0

Graham Morrison can't resist the lure of new features, even when they're from a development branch.

First, a warning. This isn't a stable release. Like lots of open source projects, *Scribus* uses the odd-minor-number versioning scheme to differentiate a development release from a stable release. The current stable release of *Scribus* is still version 1.4.5 (the 4 being the minor number we're talking about), which was made available in February to fix a few bugs.

The version we're looking at, 1.5.0, is the first major update to the development branch, and it's not in any way meant for a production environment, or for people who rely on *Scribus* to get some real work done – people like us! In particular, it's no longer compatible with the old file format, so you can't try working with 1.5.0 and revert to a stable release if you encounter a bug and need to continue with the same file. But now that we're trying to use *Scribus* more and more at the magazine, it makes sense for us to also check out the new features and see where development is headed, especially when this is already a huge update.

In particular, and like many other applications, *Scribus* has finally made the transition from *Qt 4* to *Qt 5.4*. This is long overdue and makes *Scribus* snappier and less resource hungry, at least if other *Qt* apps are anything to go by. The old *Scribus* was one of the last applications we had installed still using the old version, so it will be good to see that dependency go. As for features, the big new addition is the rewrite for tables. In the old version, tables were a simple frame where individual cells felt more like a hack. It was difficult to resize and realign cells, for example. You can now drag columns, insert and delete rows/columns and merge cells together, all options available from their own menu.



Les Arcs

Then the new features start rolling in. There are new vector tools for arcs and spirals, and the new calligraphic pen works brilliantly, especially with a stylus. There's a new picture browser plugin too, accessible from the 'Extras' menu. This enables you to browse images and tag them with your own descriptions, which can then be searched or grouped together. Considering the number of images we get through in an issue, this is going to be a great addition.

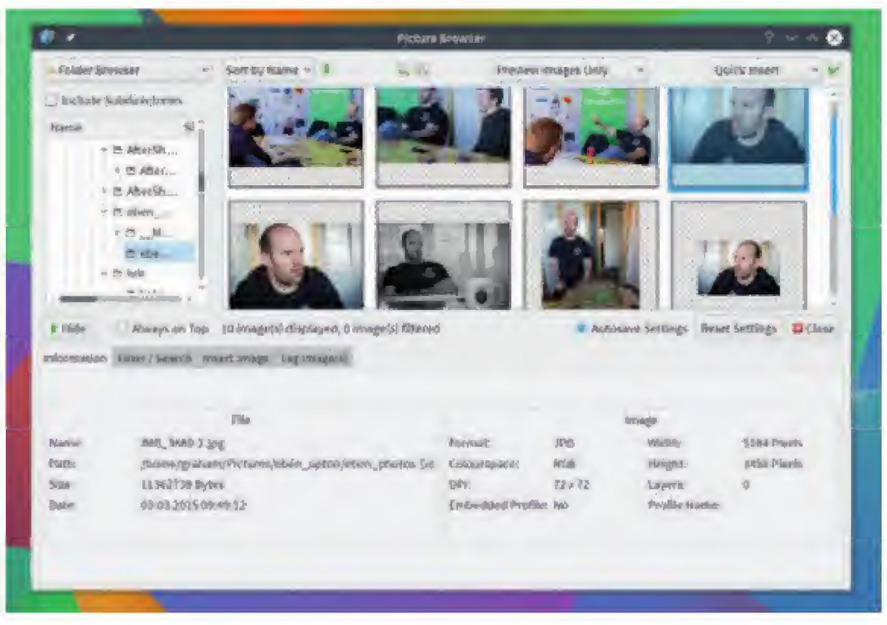
There's also been a huge amount of effort put into file compatibility, and there's a vast array of improved and new import/output filters. We tested the new PDF import, for instance, and experienced no problems with the translation (unlike with the previous version), and it was faster. We also tested some of our *Adobe Illustrator* files, and these also worked brilliantly. For the first time, *Adobe Indesign* gets an import filter, but this is only for the XML format rather than the proprietary binary format. We're hopeful that *InDesign* support will come, because we're sure this will help many designers who want to move away from their chosen platform.

As well as the new features, we also experienced great stability, despite this being a development release, although we wouldn't dare us it for production, and we hope that *Scribus* can maintain this fantastic form. We're now eagerly looking forward to *Scribus 1.6.0*, and despite there being no release schedule for this, it can't come soon enough. LV

The code to handle many import and output formats, including PDF and *Illustrator*, has been completely rewritten, with already excellent results.

DATA

Web
www.scribus.net
Developer
Scribus Team
Licence
GPL



Manage the huge number of images you typically have to deal with in print with the new Picture Browser.

LINUX VOICE VERDICT

A great development release that shows huge potential for the next stable update.

★★★★★

Building Machine Learning Systems with Python, Second Edition

Ben Everard makes computers learn so he doesn't have to

Machine Learning Systems with Python takes the reader on a tour of the *SciPy* module's machine learning routines. It doesn't dwell too much on what the algorithms do; instead it focuses far more on the practical side of things. Because it's mostly focused on using the module, the code is quite simple, so you don't need to be a particularly skilled Python programmer to follow this book, though decent school-level maths will help.

Most of the book covers the problem of classification. That is, trying to identify what class a particular piece of data should be in. The most famous classification problem is spam filtering, where a piece of software has to classify whether a particular piece of software is spam or ham (ie not spam).

Before you venture down the path of machine learning, you need to understand that it's a complex field. As well as knowing how to use the techniques, you need to know when, and the subtleties of it.

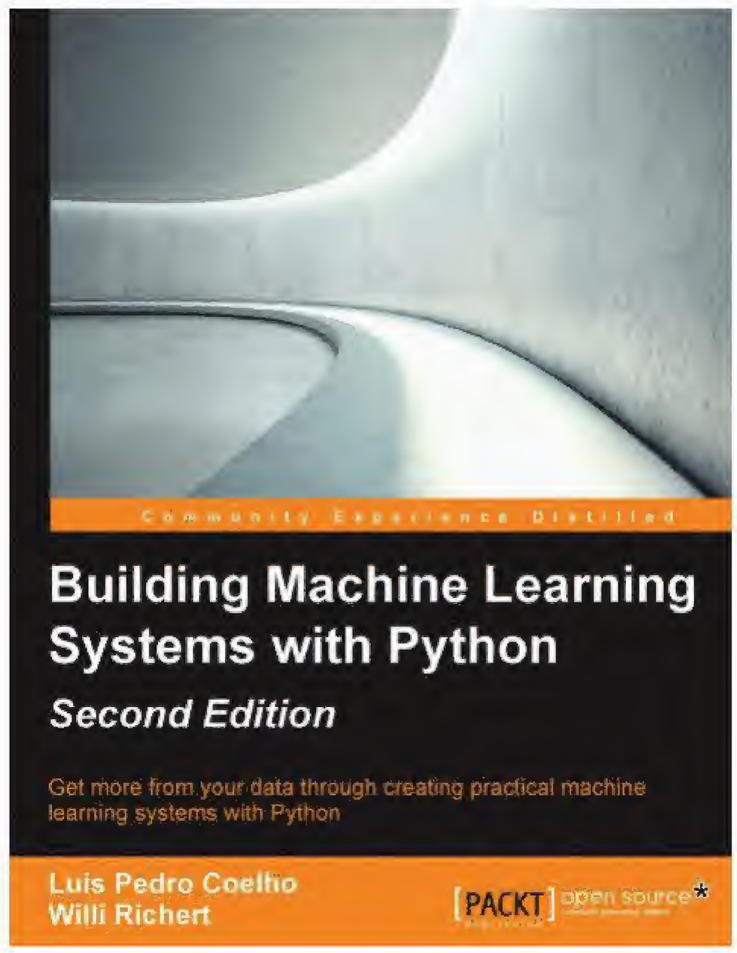
While this book gives you a good introduction, it won't make you an expert in machine learning. That's not necessarily a criticism – no sensibly sized book could take you from beginner to expert in ML. However, before embarking on a machine learning adventure, you need to be aware of the challenge.

LINUX VOICE VERDICT

Author Luis Pedro Coelho and Willi Richert
Publisher Packt Publishing
Price £32.99
ISBN 978-1784392772

A solid introduction to the basics of implementing machine learning with SciPy

★★★★☆



We long for the day when we can **import brain.**

Seven Concurrency Models

Will the spinning top eventually fall? **Graham Morrison** may have an answer.

After working on my own projects, I have some form with concurrency. I spent months attempting to coerce a dozen different multithreaded processes to listen to the same clock, and then report things at the same time. My code was left looking the opposite of the 'crystal clear, high-quality' code promised by *Seven Concurrency Models*, and even when it worked as expected, I no longer understood my own solution.

Covering seven different solutions in seven weeks (to give the book its full title) is hugely ambitious. The subject is complicated and often mind-bending, making this a book with a very specific readership. And rather than my humble threaded code, the concurrency described here is designed to scale: think Twitter rather than *tmux*. The book's great trick is that its examples and text uses functional programming, rather than procedural, to explain what needs to be done and when. It does this with Clojure, and while it borrows from things like Go's concurrency model, we'd have preferred to see Go actually used rather than name dropped. But that's our preference for what's currently a very cool language.

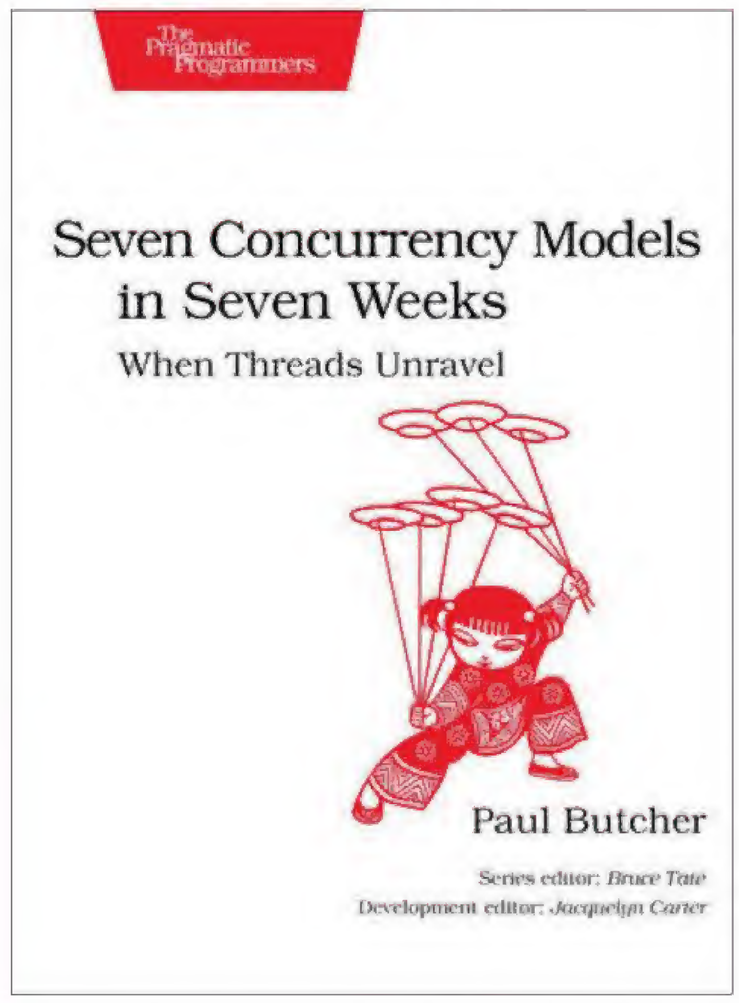
This book succeeds in teaching people with no specific knowledge of concurrency, (but firm programming skills), what an ideal solution looks like. It builds in complexity through its nine chapters but remains readable and interesting, leaving us with a much clearer idea for our own projects.

LINUX VOICE VERDICT

Author Paul Butcher
Publisher Pragmatic Bookshelf
ISBN 978-1-93778-565-9
Price £25.50

A complicated subject, but one that's all too easy to ignore without books like this.

★★★★☆



If we had the choice of any super power, it would be the ability to infinitely multitask.

The Open Organization

Graham Morrison is looking for a job at Red Hat.

All proceeds from the sale of *The Open Organization* will be donated to the EFF.

A few years ago, our esteemed colleague and Friend Of Linux Voice, Jonathan Roberts, was writing a feature about Red Hat. He was sitting at his desk, silently taking notes from a phone call, occasionally interrupting to ask a question or two. When the call was over and we all wanted to know what had kept him enthralled for so long, he said he'd been on the phone to Jim Whitehurst. "Jim Whitehurst, CEO of Red Hat?!" we 'd half asked, half shouted.

"Yes", he said. "That Jim Whitehurst. He's an awesome guy."

And here is a book by the very same Jim; eloquent, patient and readable. No other CEO from any other company could have written it. It's about how Red Hat has become incredibly successful while always doing things the open source way. "The best idea wins regardless of whether the idea comes from the top executive or a summer intern," as Jim puts it. And it's this ethos that he's put into his leadership.

We can't help but imagine what the world of business would be like if other CEOs had a similar attitude and vision towards success and innovation.

LINUX VOICE VERDICT

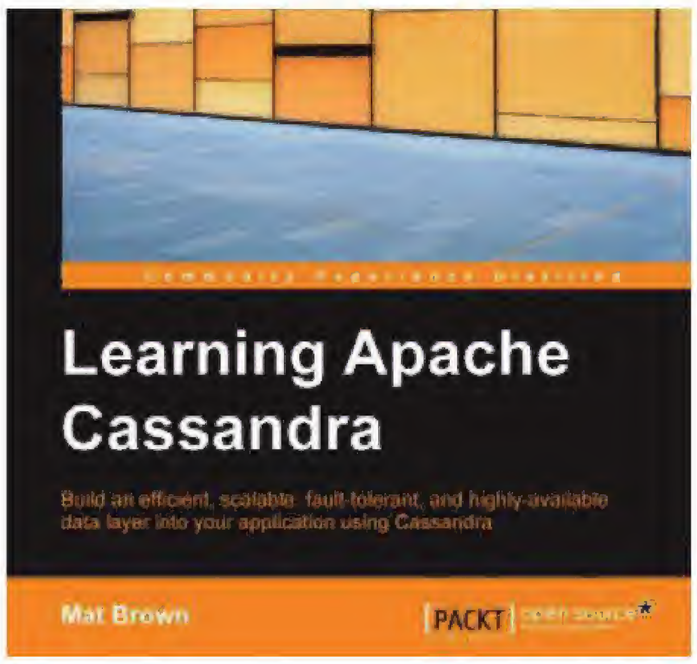
Author Jim Whitehurst
Publisher Harvard Business Review
ISBN 978-1625275271
Price £19.73

Could only be improved if Jim donated the book's proceeds to the EFF. Oh, he does.

★★★★★

Learning Apache Cassandra

Ben Everard is looking for work at either Apple, CERN, IBM or Netflix.



Cassandra was developed by Facebook.

Apache's *Cassandra* database is designed to scale horizontally. That means you don't just have a single database, you have five, or ten, or as many as you like all managing the same pool of data. If you need more space or performance, you just get another server and add it to the collection. *Cassandra* manages this through a combination of clever coding and (we're pretty sure) black magic. The important thing from a user's perspective isn't how it works, but how to get it to work, and it's this process that Matthew Brown looks at in *Learning Apache Cassandra*.

Brown takes the use through the CQL query language, which is similar to Structured Query Language (SQL), but different enough to cause problems for the uninitiated. As you work through the book, you gradually build up a real-world application that demonstrates the power of *Cassandra*, and the issues you face using it.

LINUX VOICE VERDICT

Author Matthew Brown
Publisher Packt Publishing
Price £27.99
ISBN 978-1783989201

A great book for anyone switching from a relational database to *Cassandra*.

★★★★½

ALSO RELEASED...

If you can't afford Virgin Galactic, do it yourself!

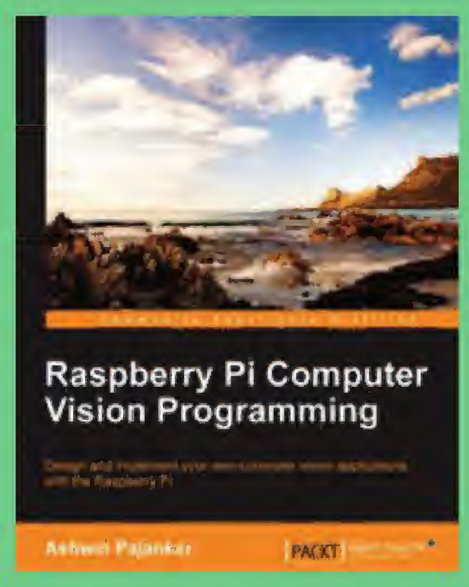
DIY Comms & Control for Amateur Space

We've had some radio components sitting in a Tupperware box for some time, so we're interested in the theory this book promises to offer on creating an uplink and a data download station for our space empire.

The Unity games engine is free to use.

Getting Started with Unity 5

The Unity programmer's games engine, used by many of the best games, is now free. And, drumroll, available for Linux. So if you've always wanted to get into games development, now is the perfect time. All you now need is a book on getting started with Unity.



The eyes have it.

Raspberry Pi Computer Vision Programming

If Andrew Conway's awesome tutorial on infrared imaging with the Raspberry Pi (see p92) has whetted your appetite for more vision projects, here's a whole book's worth that takes the same ideas further. LV

LINUX VOICE

RASPBERRY PI 2 DISTROS

GROUP TEST

Since you can now use the Raspberry Pi 2 as an everyday desktop, **Mayank Sharma** needs a distro that fits the fill.

On test

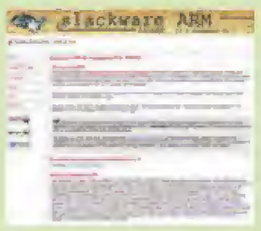
Rasbian

URL www.raspberrypi.org/downloads
VERSION 2015-05-05
DESKTOP LXDE
Can the reigning champion maintain its winning streak on the new Pi 2?

Ubuntu Mate

URL www.ubuntu-mate.org/raspberry-pi
VERSION 15.04
DESKTOP Mate
Will it extend its empire on to the mini PC as well?

SARPi2



URL http://rpi2.fatdog.eu
VERSION 13Mar15
DESKTOP KDE/Xfce
How does the grand-daddy of Linux distros perform on the PYT?

Arch Linux

URL archlinuxarm.org/platforms/armv7/broadcom/raspberry-pi-2
VERSION NA
DESKTOP NA
One of the most flexible Linux distros.

DietPi

URL http://fuzon.co.uk/phpbb/viewtopic.php?f=8&t=6
VERSION 52
DESKTOP NA
Has it managed to shed the excess?

Minibian

URL https://minibianpi.wordpress.com
VERSION 2015-02-18
DESKTOP NA
Is this the best of cholesterol-free distros for the Pi?

Raspberry Pi 2 distros

The original Raspberry Pi struck a chord with anyone who wanted a tiny little device that had enough juice for a specialised task. Thanks to desktop distributions optimised for the Pi, in particular Raspbian, you could also use the Pi as an underpowered desktop. But with the shiny new Raspberry Pi 2, the device for the hobbyist has broken into the mainstream. With a quad-core processor and 1GB of RAM, the new version has the right kind of components and physical resources to outpace some full-sized desktops produced in the last decade or so.

However, the new Raspberry Pi 2 uses a processor based on a different ARM architecture than the original Pi. Among other things, this change means that you can't use distros designed for the original Pi on the new Pi 2 straight out of the box. Since its release some months back, distros that target the Pi have been working hard to put out Pi 2-ready versions of their wares that take advantage of the extended hardware on the device.

We already know that this will be the first Pi that'll be supported by official releases from both Microsoft and Canonical, although their releases might not be what you expect. So in this group test we'll take stock of the available distro options. Instead of specialised builds, we're on the lookout for a distro that lets us use the Pi as a general purpose desktop and extends all the benefits we'd expect from a regular desktop Linux distribution.

Absent friends

The one omission we regret is Pidora, the Fedora flavoured distro for the Pi. Unfortunately the project has lost steam over the course and isn't yet available for the Pi 2. We'll also be leaving out RiscOS, which despite being an excellent OS, isn't Linux and might be unfamiliar to many of our younger readers.

"The Pi 2 will be supported by official releases from Microsoft and Canonical."

Windows 10 on the Pi 2

When Eben Upton released the Pi 2 he also announced that the Raspberry Pi Foundation would be collaborating with Microsoft to get Windows 10 on the new device. This is possibly because of Microsoft's work on Windows RT for devices that runs ARMv7, such as the one that now powers the new Pi. While the announcement broke the internet, it wasn't what many thought it to be. Getting Windows 10 to work on the Pi is part of Microsoft's program for Internet of Things [IoT] devices. The Pi version of Windows 10, which will be released at an as-yet unspecified time in the future, is meant for developing IoT apps and will probably just boot to a command line interface.

Ubuntu Snappy

Canonical wants a share of the Pi as well.

The other big announcement that accompanied the debut of Raspberry Pi 2 was the official support from Canonical. However, just like the Windows 10 version, the official Ubuntu release for the RPi 2, dubbed Ubuntu Snappy, isn't a full-fledged desktop but rather a minimal server image. Ubuntu Snappy isn't a desktop distro and is instead designed for developers to enable them to craft custom images for specific needs or for containers such as Docker.

Keeping in mind its intended goal and purpose, Snappy is also conceptually different from the other Ubuntu releases. One of the design goals of the distro is to keep the various apps isolated from one another. This means that in addition to missing a graphical desktop, the biggest difference in Ubuntu Snappy is that it doesn't use the *apt-get* package management system. Instead it manages packages with the new containerised system, which Canonical claims to be more "snappier" and gives the distro its name. The distro is still under active development and supports a limited number of packages, which you can interact with either via the command line or the custom-built web-based package manager called *WebDM*.

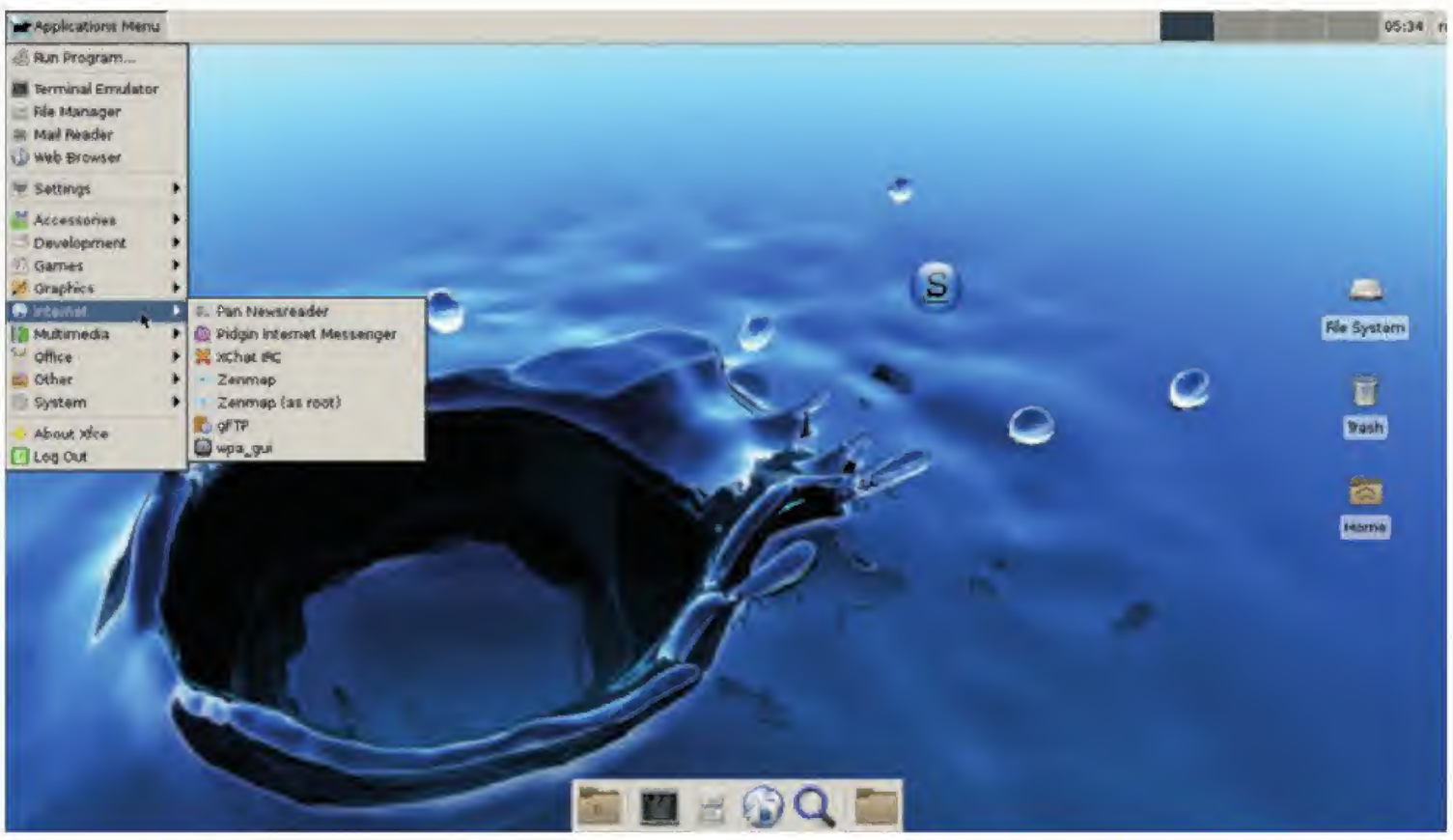
SARPi2

For the Linux aficionados.

SARPi is the name of the sub-project that produces the ARM port of the Slackware distro. Just like its desktop sibling, Slackware on the Pi is an acquired taste that won't appeal to everyone.

Unlike most other Pi distros, SARPi2 has a very involved installation process. It involves downloading a small boot image along with an optional set of packages for a complete network-less install. While for most distros the tricky bit is booting the Pi after you've transferred the disk image to the SD card, for SARPi this is just the beginning. The installation process involves laboriously navigating an *ncurses*-based menu, defining partitions, selecting packages, configuring the network and mounting an external source of packages, which takes about an hour to install depending on the source of the packages and the speed of your card.

Then you go through another round of steps manually defining the nitty gritties of the distribution. And that's just the installation. Maintaining and administering the distro involves further geekery on the CLI. For example, you'll have to edit the **mirrors** file and uncomment the entry for the mirror you wish to use before you can update or install packages.



Installing the full distro requires about 8GB of disk space, so you may need to prune the packages.

Down the rabbit hole

While this level of involvement might seem masochistic to most, Slackware users would have it no other way. Slackware doesn't make choices on behalf of its users and SARPi2 follows the simplicity-in-system-design principle of the desktop version to the letter. The good news for the non-Slackware users among us is the availability of detailed installation guides and other documentation. The SARPi2 website hosts an illustrated guide that meticulously tracks the installation process and helps you sail through the installation without any issues. Also, if you need handholding there are active Slackware forums on websites such as **LinuxQuestions.org**.

By default, SARPi2 installs the KDE desktop, but you can replace it with the lightweight *Xfce* desktop. However, in our tests, not installing the KDE desktop breaks the *Xfce* desktop as some tools such as the *wpa_gui* insist on the presence of KDE utils like *kdesu*. Also, the distro doesn't give an estimate of the installation size after you're done customising the list of packages you wish to install. The default *Xfce* desktop includes apps such as *Gimp*, *Pidgin*, *Xchat*, *GFTP*, *MPlayer* and various *Xfce* utilities. The distro lacks a graphical browser, though you can fetch one using the package manager.

We also got SARPi2 to work with the MicroNEXT wireless adapter and the RPi camera module. However, unlike with other distros, getting anything to work is a chore for a non Slackware users. Despite Linux inherently being more involved than other operating systems, there are several things that users of mainstream distros take for granted. A task such as enabling the camera module, which can be done with a single keystroke using the *raspi-config* tool under Raspbian, involves multiple trips to the forums, loading modules, installing utilities, and editing files on SARPi2. That's not a criticism of the distro, but a reflection of how good SARPi2 is at aping the behaviour of its desktop cousin.

VERDICT

Designed for Slackware users – keep this one away from beginners.

★★☆☆☆

Arch Linux

DIY on the RPi.

The venerable Arch Linux distribution has impressive support for the ARM platform and has been running on the Pi for as long as Raspbian itself. However, pretty much like the Slackware-based SARPi2 distro, Arch on the Pi sticks to the design principle of its desktop version and puts the user in charge of building their own OS virtually from scratch.

Arch Linux on the Pi isn't meant for the average desktop user. You wouldn't even be able to transfer its image onto the MicroSD card without Linux. Furthermore, you even have to partition the memory card yourself and manually copy the filesystem from the command line. And that just gets you to the Arch command line interface with nothing more than a bare-bones system with a kernel. From here you have to laboriously build your system from the ground up. But just like Slackware, Arch Linux would have it no other way.

One of the best tools in Arch is its *Pacman* package manager. With the CLI package manager you can assemble a fully functional desktop in under an hour. Yes, that might be 60 minutes more than the average desktop user would be willing to invest in setting up a desktop, but it enables you to cut the bloat on the installation. For example, setting up a Mate-based desktop with the usual collection of apps took us just over an hour. But for our troubles we got a streamlined desktop that's quick off the heels and boots into the desktop in about 15 seconds.

Arch Linux also supports the Pi-specific functions including the camera module and the GPIO pins. But once again, enabling the support for these peripherals is a more involved process than on Raspbian. However, one of the strengths of Arch is its documentation and the distro's DIY nature chimes with the Raspberry Pi's education ethos.

"Arch Linux on the Raspberry Pi isn't meant for the average desktop Linux user."

You can implement the advice on the Arch Wiki for the desktop version on the Pi installation as well.

VERDICT

Another distro that requires familiarity with its desktop version.

★★★★★

Minibian

Raspbian from scratch.

The idea behind Minibian is to create a minimal distro image with a small footprint that's fully compatible with the official Raspbian distro, using the same underpinnings as its latest release. Instead of stripping away unnecessary components from the main distro, the developer of Minibian assembles this distro from the ground up by pulling in packages from the main Raspbian repository.

According to the project's website Minibian is designed for embedded projects and makes available the maximum amount of the physical resources on the Pi. In our tests, a fresh install took slightly over 300MB on the MicroSD card and spared over 900MB of RAM. Even after installing the *LXDE* desktop, and a handful of graphical apps including the *Iceweasel* web browser and the *Synaptic* package manager, the distro used only about 111MB of RAM.

The stock Minibian image boots up to the login prompt in about 15 seconds. It ships with the DHCP daemon and the SSH server, so you can manage it remotely via the Ethernet. Since it uses the same mirrors as Raspbian, you can *apt-get* useful scripts such as *raspi-config* and *rpi-update*, though it'd be nice to have these useful scripts pre-installed. Once installed you can use these scripts to enable the camera module and work with the GPIO pins. You can also use the mirrors to install the necessary bits to get Wi-Fi to work or stuff the distro with graphical apps or even a full-fledged desktop.

But that's not the use-case its developer had in mind. Minibian is meant for anyone familiar with Debian who wants to use the Pi as a dedicated server. It's ideal for running security-related web apps, since there are no unnecessary daemons and services besides the one that you choose to install. While Minibian is a wonderful little distro, you'll have to weigh it along with its biggest competitor: the similarly positioned DietPi distro, which offers a more convenient environment for fleshing out the base distro.

Minibian isn't designed for the average desktop user, but can be converted into one with a few keystrokes.

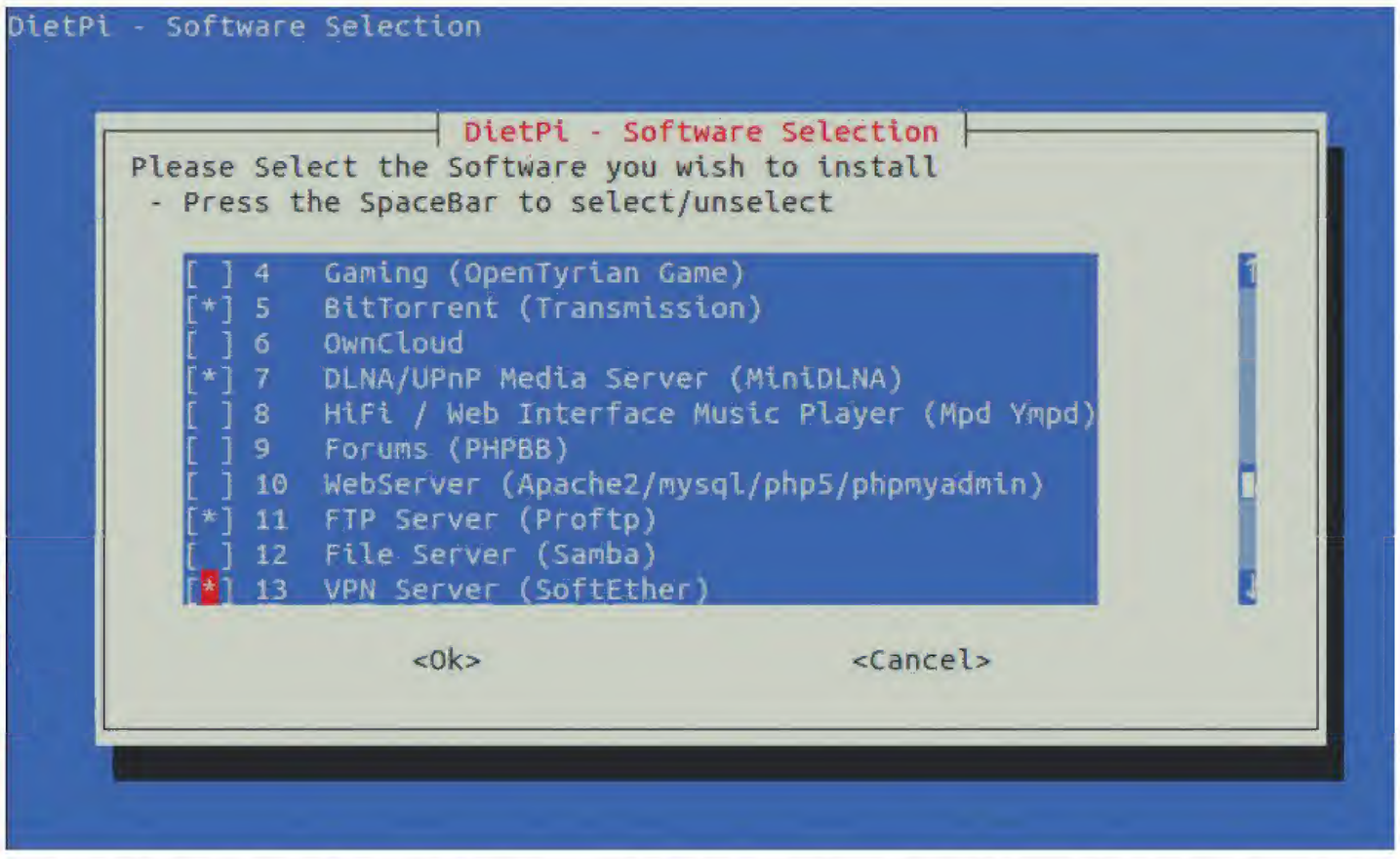
VERDICT

Pruned version of Raspbian that can be used as a regular desktop.

★★★★★

DietPi

Is sugar-free any good?



DietPi has a software installation tool, and you can also use the custom script to install additional software and resources such as a bunch of GPIO projects.

One of the most popular uses of the Raspberry Pi is as an always-on and efficient standalone server. The DietPi distro installs the bare minimum components you need to flesh out the installation according to your needs. It isn't the only distro that does so. But unlike others, DietPi goes one step further and offers a nice menu to help you pick and choose a functionality for the Pi. For example, using DietPi's custom package management script you can turn the base installation into a filesharing server, a web server, a file server, a VPN server, a seed box and even into an *LXDE*-based desktop.

The distro ships in a 7zip-compressed archive, so Linux users will need to grab *p7zip* from the repos of their distros to extract the image file. Another good thing about DietPi is that it lets you tweak its configuration by editing a text file before you boot the Pi with it. So if you plan to use a Wi-Fi adapter with your installation, enter the SSID of your Wi-Fi network and its password in the distro's configuration file after writing the image.

When you boot from the card, the distro will automatically resize itself to take over all the free space on the card and then check for updates as soon as you log in for the first time. This is good, as it takes care of the two most important aspects of using the Raspberry Pi as a server of any sort. It then launches an installation wizard to quiz you about some aspects about the installation. The most crucial information it seeks is whether you'd like to use a USB drive with the installation. If you decide to skip this you won't be able to configure one later! This is an extremely weird limitation, but at least the screen gives you verbose feedback.

Once you've configured the distro, you're taken to the software selection screen. You can exit the tool at this point, which will only install the bare minimum base along with DietPi's custom tools, using which you can flesh out the installation later. At this point you can use the distro's pre-installed SSH server to log into the installation remotely.

Easy to build on

In addition to the helpful software installation tool, DietPi includes a custom configuration script for managing various aspects of the Raspberry Pi and the connected hardware. You can use the script to overclock the Pi, change the resolution, mount remote shares, and even enable the Pi camera module. It also includes a tool for benchmarking the performance of the Micro SD card and any connected USB drives.

VERDICT

The best minuscule distro with the right kind of tools for easily fleshing it out.

★★★★½

Specialised builds

Bespoke distributions for your Raspberry Pi.

In addition to the mainstream distros tested in this feature, there are several purpose-built distros available for the Raspberry Pi 2 as well. You can use the HDMI port on the Pi to connect it to your HDTV and use it to power your home theatre. Distros like OpenELEC and the upcoming OSMC (Open Source Media Centre) wrap the popular *Kodi* media player into dedicated ready-to-use home theatre PC (HTPC) appliance. Using these distros you can easily move media inside your Pi-powered HTPC and control playback remotely.

Sticking with multimedia, you can turn your Pi into the ultimate jukebox with the Pi MusicBox distro. The distro can handle all sorts of media files stored locally and over the network and can also fetch music from streaming services like Spotify, Google Play Music, SoundCloud, Last.FM and more. The distro interfaces with various desktop and mobile clients that you can use to control playback. Furthermore, Pi MusicBox can also convert the Pi into a DLNA compatible device that can stream music from other DLNA devices (DNLA is a manufacturers' 'standard' for sharing data over a home network).

If you're a fan of retro gaming, RetroPie will give you access to every open source gaming emulator on the planet and includes drivers to let you hook up modern day gaming controllers. And if you ever need storage space, get hold of a couple of large capacity disks and hook them to the Pi, which you can then use as a power-efficient network attached storage device with the OpenMediaVault distro.

Set up a home NAS with OpenMediaVault.

Raspbian vs Ubuntu

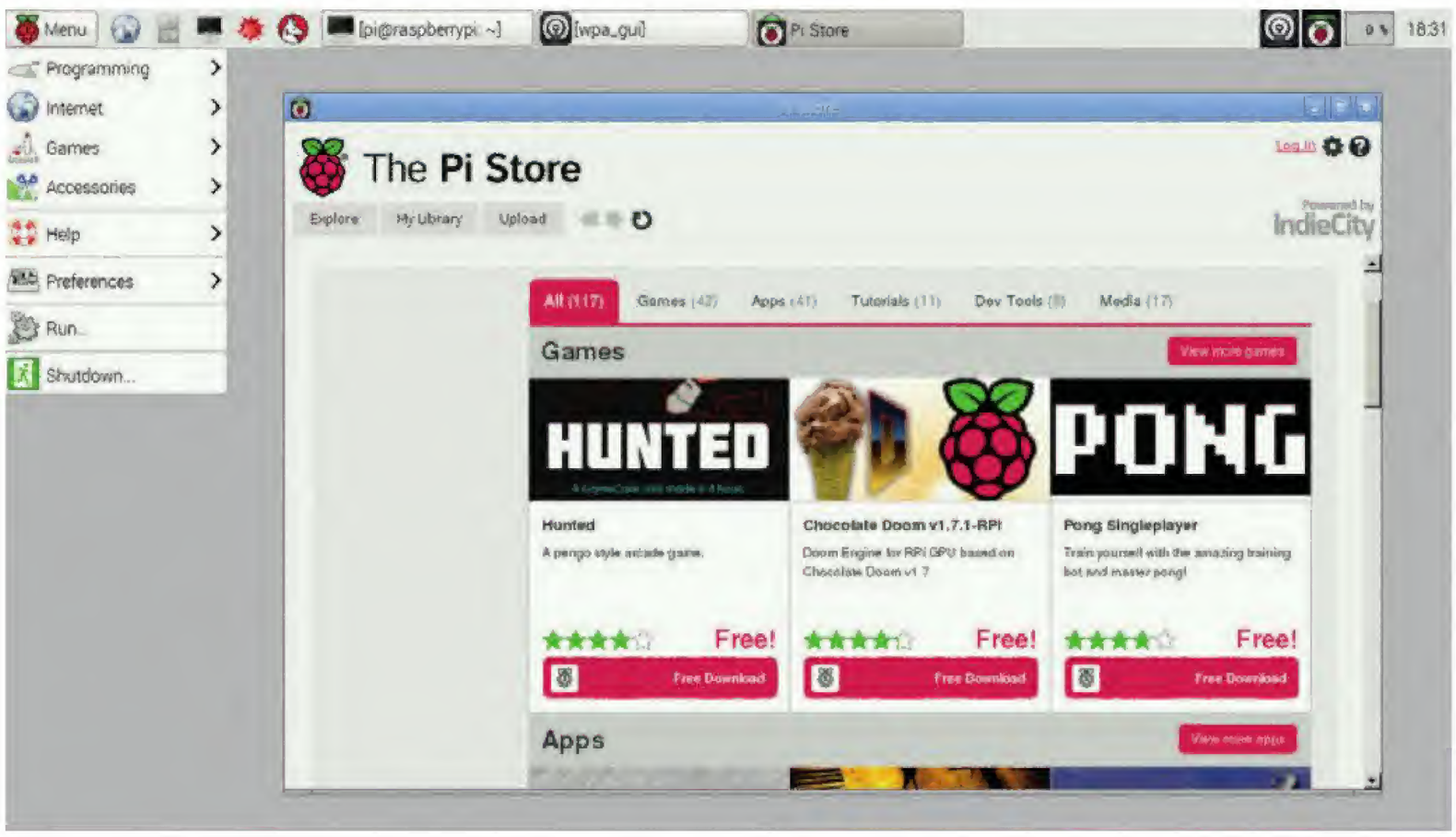
The fight of the century.

With its first release in 2012, Raspbian is one of the oldest distros for the Raspberry Pi that runs on both the original Pi and the Pi 2. On the other hand, Ubuntu is a greenhorn making its debut on the tiny PC thanks to its ARMv7 chip.

Due to its age-old support and strong foundation, Raspbian is recognised as the recommended distro for the Pi. It's also the easier to install of the two thanks to it being part of the Pi's NOOBS installation mechanism. Raspbian uses the *LXDE* desktop and its default selection of a handful of apps is tailored for young audiences, particularly those who want to hone their programming skills. There's *Sonic Pi*, *Scratch*, *Minecraft*, *Wolfram Language* & *Mathematica* along with interpreters for both Python 2 and 3.

While these apps make it a wonderful starter kit for educational purposes, it fails miserably as a regular desktop. The included web browser is good for reading HTML documentation but can do little else as it ships with no plugins. Three apps that are of note are the graphical app for configuring wireless adapters, the Pi Store client that pulls in apps from **store.raspberrypi.com** and the *ncurses*-based *raspi-config* script.

You can think of *raspi-config* as the BIOS for the Raspberry Pi. It helps you tweak the hardware of the Pi; for example, changing its clock speed and enabling the camera. On first boot, Raspbian launches the script to enable you to change the password for the default user and expand the distro to take over the entire card among other things. If you're willing to put in some time you can transform Raspbian for everyday use thanks to the gazillions of apps at your disposal via its mirrors.



Raspbian boots to the desktop in about 21 seconds and leaves about 748MB RAM.

Hiya Mate!

In contrast to the blandish Raspbian, Ubuntu Mate boots into a desktop that's chock full of apps. As its name suggests, the distro is based on the Gnome 2-inspired Mate desktop and includes quite a few of its default lightweight apps as well. What's surprising though is the inclusion of feature-rich mainstream apps such as *Pidgin*, *Thunderbird*, *Rhythmbox*, *VLC*, *Firefox* and even *LibreOffice*! The inclusion of *LibreOffice* might seems asinine at first (it did to us) but the fact that it starts up in under 10 seconds is a testament to the Pi 2's processing superiority over its predecessor. Software management is handled by *Ubuntu Software Centre*, which, like the other heavyweights, performs well.

But all these apps take a toll on the distro's boot times (about 50 seconds to boot to the desktop). Ubuntu also lacks the helpful *raspi-config* script, so you have to configure extras manually. For example, you can use the camera module on Ubuntu after appending a couple of lines in the **config.txt** file. Also, GPIO works out of the box

Unlike other distros, Ubuntu Mate includes a four-step installation wizard that helps you create a user account. The distro also picks up the attached Wi-Fi adapter, though you'll have to install the OpenSSH server if you want to manage the installation remotely.

For a smooth video playback experience, either purchase the hardware accelerated plugins from the Raspberry Pi store or use the included *OMXPlayer*.

VERDICT

RASPBIAN The recommended distro for the Pi is a good desktop distro. ★★★★☆

UBUNTU MATE Chock-full of apps and performs admirably well. ★★★★☆

OUR VERDICT

Raspberry Pi 2 Distros

We aren't kidding when we say that we can find a use for each one of these distros. SARPi, Arch, DietPi and Minibian are all excellent choices for building headless servers, depending on your familiarity with their respective base distros. These distributions ship with a bare minimum base and give you complete autonomy over their package management. You can use them to take full advantage of the Pi's minuscule physical dimension to build a low footprint server that'll fit anywhere.

Furthermore, since SARPi and Arch mimic their desktop variants, you can also use them to learn and understand the workings of the geekier Linux distributions without exposing them to your regular desktop.

Between the Debian-based mini distros, Minibian and DietPi, we favour the latter for its software management scripts. They are easy to use and help lower the entry barrier making it possible for even an inexperienced user to transform the Pi into a server without messing with the CLI. This isn't how you'd ideally set up a web server for your company, but is ideal for deploying bite-sized home-based projects such as a MiniDLNA server or a seed box.

It's about the desktop, stupid

However, this group test is about finding a regular desktop distro that takes advantage of the pumped-up Raspberry Pi 2. Raspbian has done a commendable job as a desktop distro for the original Pi, which is also why it's the recommended flavour. That said, we'd like to award this test to the latest entrant on the Raspberry Pi's download page, Ubuntu Mate. The distro has just had its first release on the platform, but it comes from a strong lineage and is backed by an official Ubuntu spin. It also lacks the convenience of the *raspi-config* script, but all the features of the script are available as long as you know where to look; plus, a version of *raspi-config* is on the project's to-do list, and knowing the rate at which Ubuntu works you may find this feature implemented by the time you read this.

Ubuntu Mate on the Pi 2 is the sincerest attempt to ship a ready-to-use desktop distro for the Pi. The distro doesn't require a trip to the package repository and can be put to use straight after its desktop-style installation, which earns it the top spot.

"Ubuntu Mate is the sincerest attampt to ship a ready-to-use desktop for the Pi."

For a speed boost, take advantage of Ubuntu Mate's chassis and install a lighter desktop environment such as **lubuntu-desktop**.

1st Ubuntu Mate

Version 15.04 **Desktop** Mate

www.ubuntu-mate.org/raspberry-pi
The best distro for using the Pi as a full fledged regular desktop.

2nd Raspbian

Version 2015-05-05 **Desktop** LXDE

www.raspberrypi.org/downloads
The Pi Foundation's recommended distro is a wonderful Linux OS for educational purposes.

3rd DietPi

Version 52 **Desktop** NA

http://fuzon.co.uk/phpbb/viewtopic.php?f=8&t=6
The best option for rolling out servers and network services over the Raspberry Pi.

4th Arch Linux

Version NA **Desktop** NA

http://archlinuxarm.org/platforms/armv7/broadcom/raspberry-pi-2
One of the two distros that'll appeal to existing users of its desktop flavour.

5th SARPi2

Version 13Mar15 **Desktop** KDE/Xfce

http://rpi2.fatdog.eu
Its installation process is just too involved to appeal to everyday desktop users.

6th Minibian

Version 2015-02-18 **Desktop** NA

https://minibianpi.wordpress.com
A minuscule distro that offers no incentives over its closest rival.

	Installation	Desktop	Based On	Pi models supported
Raspbian	Simplest	LXDE	Debian Wheezy	Pi 1, Pi 2
Ubuntu Mate	Straightforward	Mate	Ubuntu 15.01	Pi 2
SARPi	Complex	KDE/Xfce	Slackware	Pi 1, Pi 2
Arch Linux	Complex	-	Arch Linux	Pi 1, Pi 2
Minibian	Involved	-	Raspbian	Pi 1, Pi 2
DietPi	Straightforward	-	Raspbian	Pi 1, Pi 2

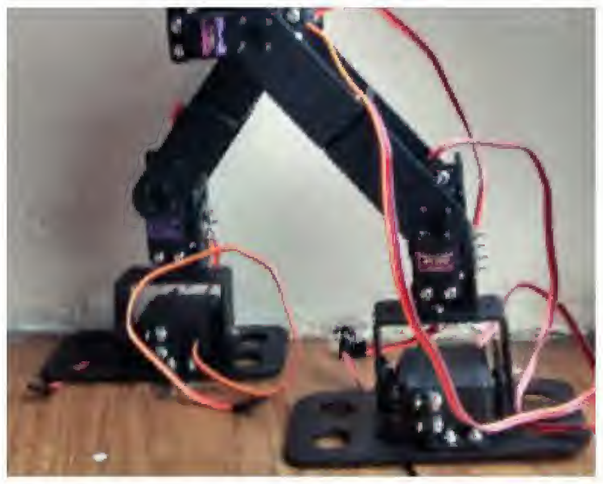
NEXT MONTH IN LINUX VOICE

ON SALE THURSDAY 30 JULY

GEEK UP YOUR SUMMER

Enhance your summer with our clutch of things to make and do – all powered by Linux and Free Software, of course

EVEN MORE AWESOME!



Robots!

Ben has been locked in the shed with his soldering iron for a while. We're not sure what he's building, but the shareholders of Boston Dynamics are looking worried...

Inside ORG

The Open Rights Group do good things on our behalf; lobbying politicians, campaigning and fighting the good fight. Here's what they're up to now.

Drupal

If your website is anything more than a static HTML page, you probably need a content management system – and *Drupal* is one of the best there is.

LINUX VOICE IS BROUGHT TO YOU BY

Editor Graham Morrison
graham@linuxvoice.com
Deputy editor Andrew Gregory
andrew@linuxvoice.com
Technical editor Ben Everard
ben@linuxvoice.com
Editor at large Mike Saunders
mike@linuxvoice.com
Creative director Stacey Black
stacey@linuxvoice.com

Editorial consultant Nick Veitch
nick@linuxvoice.com

All code printed in this magazine is licensed under the GNU GPLv3

Printed in the UK by
Acorn Web Offset Ltd

Distributed by Marketforce (UK) Ltd, Blue Fin Building, 110 Southwark Street, London, SE1 0SU
Tel: +44 (0) 20 3148 3300

Circulation Marketing by Intermedia Brand Marketing Ltd, registered office North Quay House, Sutton Harbour, Plymouth PL4 0RA
Tel: 01737 852166

ISSN 2054-3778

Subscribe: **shop.linuxvoice.com**
subscriptions@linuxvoice.com

Valentine Sinitsyn develops high-loaded services and teaches students completely unrelated subjects. He also has a KDE developer account that he's never really used.

CORE TECHNOLOGY

Prise the back off Linux and find out what really makes it tick.

Non-trivial Iptables

You know how to compose basic iptables rules – now take them further with these clever tricks.

Like any self-respecting operating system, Linux comes with the built-in firewall. If this statement makes you think of *iptables*, or Xtables in general (which refers to *iptables*, *ip6tables* etc), you're right. However, that's only the tip of the iceberg. *Iptables* is a userspace tool that relies on an in-kernel framework called *Netfilter*. The latter is what hooks into network subsystem, analyses packets as they come in and out and acts accordingly.

There are numerous good tutorials and howtos on *iptables* (see **http://netfilter.org/documentation**), and we're not going to repeat them here. What we are going to do is to shed some light on lesser known, more obscure features you may find useful in real-world scenarios.

A firewall is basically a set of rules containing conditions to select packets (or "matches", in *iptables* parlance), and actions (or "targets") to take on them. Target names in *iptables* are, by convention, uppercase. For instance, you can DROP (or silently discard) all ICMP type 8 (Echo Request) messages, and make your host invisible to "ping scans". This is not recommended though, as ICMP is not just a ping workhorse but an essential protocol for networks to run smoothly. If you block it blindly, you're almost certain to run into obscure bugs.

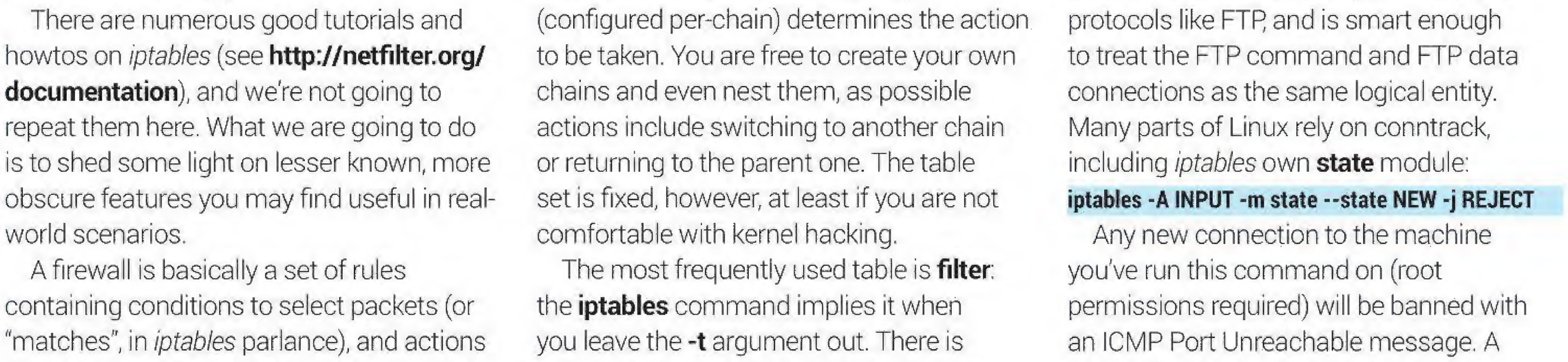
In *iptables*, rules form chains that are grouped into tables (hence the name). Packets traverse them in a predefined order (see the diagram below). If a packet matches no rule, the default policy (configured per-chain) determines the action to be taken. You are free to create your own chains and even nest them, as possible actions include switching to another chain or returning to the parent one. The table set is fixed, however, at least if you are not comfortable with kernel hacking.

The most frequently used table is **filter**: the **iptables** command implies it when you leave the **-t** argument out. There is also **mangle** for setting marks (covered shortly), and **nat** to perform network address translations. A relative newcomer to this family, **raw** is traversed before conntrack comes into play, and you can use it to prevent connections from being tracked via the **NOTRACK** target.

What's conntrack, you ask? It's a part of *Netfilter* (officially the *Netfilter* connection tracking module) that tracks network packets and determines which connection they are part of. With stream-oriented protocols like TCP it's relatively straightforward. However, conntrack also knows about some application-level protocols like FTP, and is smart enough to treat the FTP command and FTP data connections as the same logical entity. Many parts of Linux rely on conntrack, including *iptables* own **state** module:

```
iptables -A INPUT -m state --state NEW -j REJECT
```

Any new connection to the machine you've run this command on (root permissions required) will be banned with an ICMP Port Unreachable message. A connecting party will probably receive a "Connection refused" error. All existing connections should continue, so you won't shut down your own SSH session, for



This diagram shows how network packets traverse built-in *iptables* chains and tables. Note that *Netfilter* is only part of the story.

instance. Remember though that changing firewall rules over SSH is almost certainly a bad idea.

Connection tracking not only provides states (so this type of firewall is called "stateful"). It also maintains arbitrary 32-bit integer marks associated with connections. These are known as "connmarks". To set a connmark, do the following:

```
iptables -t mangle -A PREROUTING -p tcp --dport 80 -j CONNMARK --set-mark 0x1
```

The **mangle** table is a traditional place to set marks. Here, we do it in the **PREROUTING** chain, or prior to routing decisions. All TCP traffic targeting port 80 (presumably, HTTP) is assigned connmark 0x1. Here's how to match against the connmark:

```
iptables -A INPUT -m connmark --mark 0x1/0xff -j ACCEPT
```

0x1 is the target mark value, **0xff** is a mask. A mask is how you define bits to consider when matching; here, only the lowest 8 bits are taken into account. Masks are quite common in *Netfilter*, and are often used to effectively combine several marks in one.

While you can check marks in *iptables*, they are mainly useful for advanced routing or traffic shaping (QoS). However, tools like **ip** or **tc** can't work on connmarks directly. Instead, they rely on per-packet marks. These are different from connmarks, but you can synchronise their values with:

```
iptables -t mangle -A PREROUTING -j CONNMARK --restore-mark
iptables -t mangle -A POSTROUTING -j CONNMARK --save-mark
```

The first command copies a connection mark to the packet, and the second does the opposite.

It is also perfectly legal to set or check "plain" marks manually like this:

```
iptables -A INPUT -p tcp --dport 80 -j MARK 0x1 --set-mark 0x1
iptables -A INPUT -m mark --mark 0x1/0xff
```

Packet marks are available even if your kernel was compiled without *Netfilter* connection tracking support.

Digging deeper

Xtables sports many matchers, including third-party extensions. However, most of them work only on the network packet header, and there will be times you'd want to peek into data payload. You may want to be sure the packet targeting port 80/tcp is really HTTP, or filter DNS requests by names they contain. These times, you'll need Deep Packet Inspection, or DPI.

DPI techniques are complex and performance-hungry. You don't want them unless absolutely necessary, and we'll cover some alternatives shortly. However if you find yourself looking for a way to block or prioritise *Skype*, BitTorrent or another tricky protocol that was designed to be hard to firewall, DPI is the answer.

DPI engines aren't naively parsing all traffic coming through. Instead, they look for specific traits or signatures that uniquely identify the protocol. It is similar to how antiviruses work, and "false positives" (or misdetection) can happen here as well. Finding a good signature is a tough research problem, that takes time and money to solve, so the DPI market is dominated by proprietary (and very expensive) solutions. A German-based company named Ipoque (**www.ipoque.de**) once open-sourced a stripped-down version of its *Pace* DPI engine and built the *OpenDPI* project around it. Unfortunately, things didn't go well and the OpenDPI project was shut down in 2012. The good news is that the guys at the *ntop* project forked *OpenDPI* as *nDPI* and develop it today. It's available from **www.ntop.org/products/ndpi** under LGPLv3.

"Remember that changing firewall rules over SSH is almost certainly a bad idea."

While *nDPI* is a userspace library, it is written in portable C suitable for kernel space execution. There have also been numerous attempts to wrap it as an Xtables extension. They seem to die and rise from ashes quite regularly, so finding one that's steadily maintained is not trivial. My own favourite lives at **http://devel.aanet.ru/ndpi**.

The OpenDPI project was shut down three years ago, but development continues at **ntop.org**.

The homepage is in Russian, but you should find the download link easily: look for the topmost **nDPI-something.tar.gz**; for now, it is **nDPI-1.5.1.r9249.tar.gz**. The **rXXX** part is the *nDPI* SVN revision that the extension is bundled with.

Once you have the tarball, unpack it and **cd** into the **ndpi-netfilter** directory under top-level **nDPI-...**. Now, run **make**: you'll need the kernel and *iptables* headers installed on your machine. They are usually called **linux-headers** and **iptables-dev** in your package manager.

Wait for the build process to finish. Check there were no errors, then copy **ipt/libxt_ndpi.so** to wherever your system stores Xtables extensions (usually **/lib/xtables** or **/usr/lib/iptables**). Now, **insmod src/xt_ndpi.ko** (as root). If this complains about an unknown symbol, make sure you've also loaded **nf_conntrack** and the **x_tables** kernel modules.

The extension provides both a match and a target. You can see the options available with **iptables -m npdi --help**. Consider piping it to **less** as an **ndpi** match provides a command line switch per protocol, and the list of supported protocols is quite lengthy.

This is how to block *Skype* on your router:

```
iptables -m ndpi -A FORWARD --skype -j DROP
```

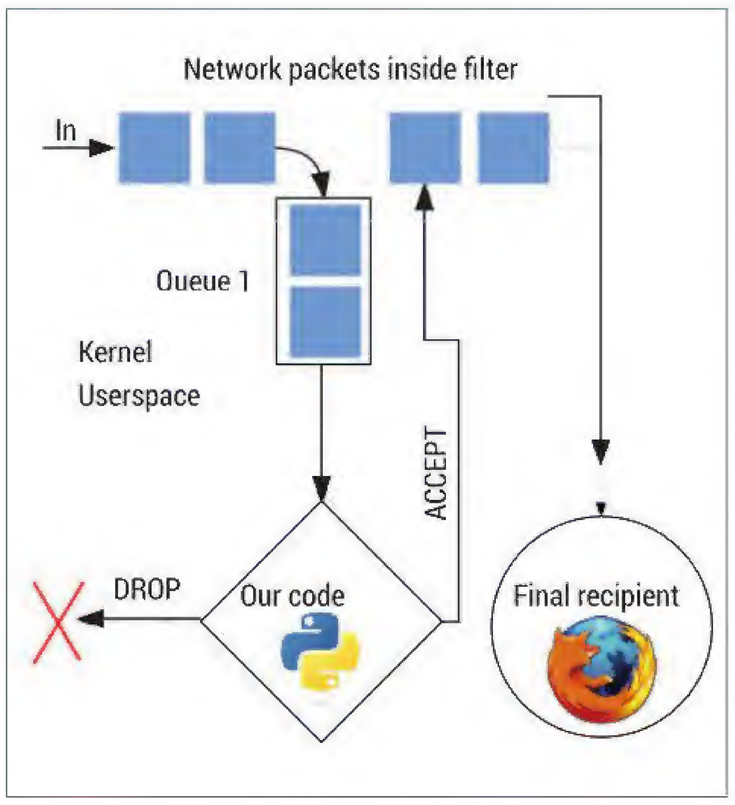
Considering possible false positives, you may decide not to block *Skype* completely but force it to use limited bandwidth. This is straightforward: assign a mark instead of **DROP** and use **tc** for shaping. You may readily convert the rule above to **-j MARK**, but there is better approach.

The *nDPI* Xtables module provides an **NDPI** target that automatically assigns marks or priority to packets according to the protocol detected. Mark values and masks are stored in **/proc/net/xt_ndpi/proto** – read this file to get the current settings in simple tabbed format. The column named **id** stores the identifier that *nDPI* assigns to supported protocols; **mark** and **~mask** are mark and mask values (negation **~** is a typo); and the last column contains a short protocol

Debugging aids

As the size of your *iptables* ruleset grows, debugging it may become troublesome. Luckily, there is one little tool to make the process easier. The **TRACE** target forces *Netfilter* to log every rule the packet traverses. This is useful only if **-j TARGET** is the first action taken on the packet; that's probably why **TRACK** is valid only in the **raw** table. For each match, you get the table and chain names, and the rule number. If the packet reaches the end of the chain or the default policy-defined action is taken, you also get a note.

To use **TRACK** you'll also need the **ipt_LOG** kernel module loaded. Traces are viewable in **dmesg** and **/var/log/kern.log** or similar, depending on your logger settings.



This is how the *Netfilter* queue operates: packets are evicted from their normal flow and re-injected as needed.

name. Two special identifiers, **all** and **any**, are wildcards: **any** stands for any recognised protocol, and **all** includes unknown ones.

By default, mark values are equal to protocol IDs, but you can write to this file to change marks. For example, this is how you assign the **0xdeadbeaf** mark to *Skype* (protocol ID **7d**):

```
echo '7d deadbeaf/ffffffff' > /proc/net/xt_ndpi/proto
```

The ID, mark and mask values must be hexadecimal numbers. We'll see further examples of NDPI usage in the later section.

Quick filters

It would be unfair to say that *iptables* can't work with network packets at byte level. At least two modules, **strings** and **u32**, come bundled just for these purposes. With **strings**, you can search for given substring (either ASCII or hexadecimal) in the packet. **u32** sports C-like expressions to check byte values at given offsets, and can even perform some bit operations like shifts. Both do their jobs well, but are somewhat limited and don't account for all possible scenarios. **u32** rules are also not easy to read, unless you have a trained eye.

The alternative comes in the form of *Berkeley Packet Filters*, or *BPF*. Despite the name revealing its BSD origins, this is the *de facto* standard technology for advanced packet filtering in Unix. *BPF* is available to many operating systems now, including Linux, of course. The primary design goal behind *BPF* was socket-level filtering for network sniffers such as *Wireshark*. This was later extended to *seccomp*, a Linux-specific sandboxing technology.

It is quite possible you've already used *BPF* filters without even knowing it. Does this looks familiar?

```
tcpdump -i eth0 udp port 53
```

This is how you limit **tcpdump**, and in fact any other **libpcap**-based program's output to DNS traffic. Internally, **libpcap** compiles this filter into *BPF*.

By itself, *BPF* is an assembler-like language that is executed in a virtual machine. There are no backjumps, so the language isn't Turing-complete, but this is a guarantee that a *BPF* program will not loop forever. Here's how a simple *BPF* program may look:

```
ldh [12]
jne #0x806, drop
ret #-1
drop: ret #0
```

This loads the EtherType field (byte offset 12) and compares it against **0x806** (the ARP protocol type). If the values aren't equal, execution continues at **drop** and returns 0. Each *BPF* program yields a number of bytes to keep from the packet, so this means discarding it completely. Otherwise, **-1** (or unsigned 65535, a maximum packet size) is returned, and the packet continues untruncated (or is accepted). You can find more examples in the **bpfc(1)** man page, which is a *BPF* assembler from the **netsniff-ng** toolkit.

> **"Berkeley Packet Filters is the de facto standard for advanced packet filtering."**

Starting at version 3.0, the Linux kernel can JIT (Just In Time) compile *BPF* filters to native machine code. To enable this feature, just do **echo 1 >/proc/sys/net/core/bpf_jit_enable**. This makes *BPF* filters really fast beasts. CloudFlare, which provides a distributed nameserver system, reported that it was able to filter 41 billion malicious DNS requests by names they contained overnight. Note that this result is obviously hardware-dependent.

How do you make use of *BPF* filters in your own ruleset, you ask? Don't be afraid, being able to program at assembler level is not a strict requirement. Still, for those who find it fun, our ASM School (which started in LV012) provides just enough background to get started.

The trick is to call the high-level syntax compiler provided by *libpcap*. Some may advise you to use **tcpdump -ddd**, which prints *BPF* opcodes, but this doesn't seem to work with newer *iptables* anymore. Better to stick with **nfbpf_compile**, which comes bundled with *iptables*. The only inconvenience is that it is built only if *iptables* is configured with **--enable-bpf-compiler**, and this is not what most Linux distributions do. So, you may need to grab the sources and recompile them yourself.

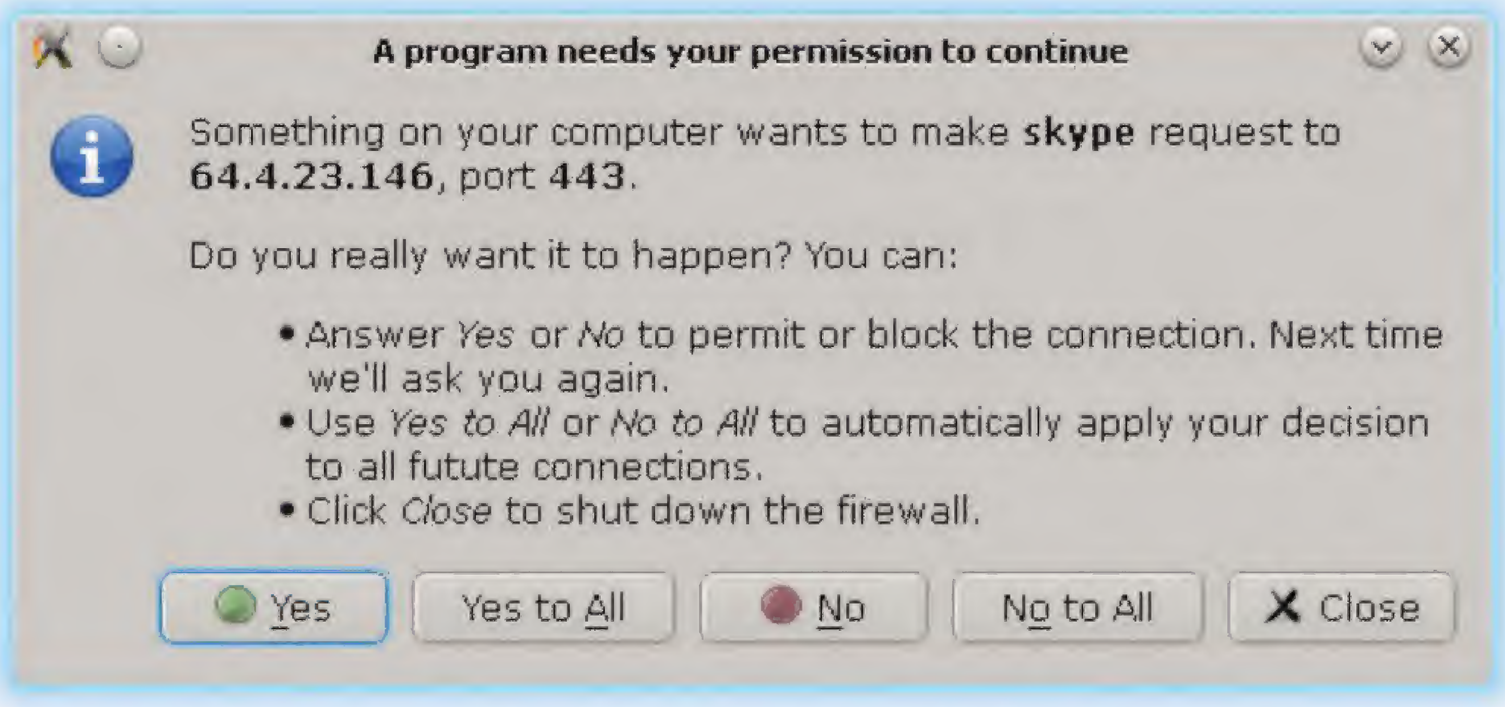
If you want to filter NTP requests, this is possible with *nDPI*, but *BPF* will probably be faster. NTP uses port 123/udp. Moreover, as per RFC958, bits 2..7 in the packet payload are set to a value less than four (this is basically how *nDPI* detects NTP, too). Call **xt_bpf** as follows:

```
iptables -A INPUT -m bpf --bytecode "$(nfbpf_
compile RAW 'udp port 123 and ((udp[8] & 0x38)
>> 3) <= 4')" -j LOG
```

Now all NTP requests will end up in your kernel log.

Deciding in userspace

At the very end, you may want some really convoluted logic, not easily expressible even in terms of *BPF*. Wouldn't it be good to have a whole C (or even Python) function that decides on a packet's destiny?



Our toy firewall can block *Skype* already. Be prepared to click "No" many times, as it will eagerly look for a way out.

That's exactly what the *Netfilter* queue (or *nfqueue* for short) is for. It's like *libpcap* in that it copies network packets to userspace. The difference is that with *libpcap* you can only see what's in the wire, while *nfqueue* lets you mangle packets' headers and data, accept them and even drop them at your own discretion. This comes in handy in some cases, and Intrusion Prevention Systems (IPS) like *Snort* or *Suricata* rely on this mechanism to implement so called "inline mode".

The **libnetfilter_queue(3)** man page describes many options available with *nfqueue*. You can choose to get only packet metadata which is faster, if you only need to check marks, for instance. You can retrieve packets partially or fully, change headers, data payloads and even metadata (including firewall marks), and you can set your verdict, like **NF_ACCEPT** to accept a packet or **NF_DROP** to discard it. Note that there's no easy way to accept a packet and continue the current chain, however.

To queue packets for userspace, we use the **NFQUEUE** *iptables* target. There are 64K queues available in Linux; **--queue-num** select the one you want. The kernel queues up to 1024 packets by default; excess ones will be discarded silently, or accepted, if the queue was opened with the **NFQA_CFG_F_FAIL_OPEN** flag. Packet loss will also occur if no program processes the queue in userspace: use **--queue-bypass** to bypass queuing in such cases. In short, be careful when working with *nfqueue*: your code is a part of the Linux networking core.

Despite all its advantages, *nfqueue* is quite slow. First, it needs to copy packets to and from userspace. *Netfilter* uses Netlink sockets (the **AF_NETLINK** family, somewhat like **AF_UNIX** we discussed back in LV015) for this purpose. Then, processing is delayed until the kernel scheduler gives your program a chance to run. All of this adds measurable overhead. As a quick test, I tried a single rule that accepts ICMP Echo requests on **lo** in the kernel (**-j ACCEPT**) and in userspace with *nfqueue*. On 32K pings, the latter is 2.9 times slower with C code and 3.8 times slower with Python (packets are fully copied). Your mileage may vary, but you can use these figures as a guide. That's the price to be paid for being able to program in userspace, and even use a high-level language like Python.

For a complete example, let's write a toy interactive GUI firewall. These things are quite popular in Windows. First of all, set up *iptables* as follows:

```
iptables -N userfw
iptables -A userfw -j NFQUEUE --queue-num 1
iptables -A OUTPUT -m ndpi --all -j NDPI --ndpi-id
--set-mark
iptables -A OUTPUT -m state --state NEW -j userfw
```

Here, we create a **userfw** chain and pass all new outgoing traffic through it. Every packet in **userfw** gets queued in queue 1. We also use an NDPI target to classify packets before queuing them for userspace.

All we need now is a Python program that reads packets from the queue, and asks the user if it is OK to let them out. A stripped-down version of this code may look like this:

```
QUEUE_NUM = 1
def handle_packet(payload):
  mark = payload.get_nfmark()
  data = payload.get_data()
  packet = ip.IP(data)
  # Get dst and dport from packet
  req_proto = ndpi_proto(mark) if mark else
'unknown'
  if ask_user(dst, dport, req_proto):
    payload.set_verdict(nfqueue.NF_ACCEPT)
  else:
    payload.set_verdict(nfqueue.NF_DROP)
queue = nfqueue.queue()
queue.set_callback(handle_packet)
queue.fast_open(QUEUE_NUM, AF_INET)
queue.try_run()
```

The **ndpi_proto()** function (not shown here) reads **/proc/net/xt_ndpi/proto** and gets the protocol name for a 'mark' that *nDPI* sets for us. Only the first packet in a connection reaches our firewall, so detection may be inaccurate. We use **dpkt** to parse network protocols; **scapy** is also a reasonable choice. You will find complete source at **www.linuxvoice.com**.

Note this is really a toy, albeit one that runs with root permissions. Modern applications make many simultaneous requests. You may not like it, and they may not like you banning these connections randomly. So, play wisely, and stop programs having valuable data before trying this.

nftables: iptables reloaded

iptables were here for more than fifteen years. They are great, but they're starting to show their age. The project to redesign iptables is underway, and preview releases are available with Linux 3.13 and up. That's what nftables are.

Nftables replace the whole family of tools (`iptables`, `ip6tables`, `ebtables` etc) we have now with only one: `nftables`. The syntax is also changed to feel more natural:

```
nft add rule filter output ip daddr 1.2.3.4 counter
```

Both tables and chains are now user-definable. Packet counters (sometimes a bottleneck in iptables) are optional, and rule may have more than one target. Nftables relies on new in-kernel infrastructure that provides optimized data structures (much like ipset) to match packets faster. For migration path, there is a compatibility layer to run iptables or ip6tables on top of nftables infrastructure.

nftables look very promising - please drop us a line if you want to know more about them. Eventually, they will replace iptables in your favourite Linux flavour, so keep an eye on the progress.

Command of the month: ipset

Quite often, *iptables* rules contain many IP addresses (think of *Fail2ban*). You can use a single rule per address, but the more rules you have, the longer it takes to match packets against all of them. What should you do?

ipset comes to rescue. It refers to both the in-kernel framework and the userspace utility to manage possibly disjointed sets of addresses and ports effectively. Sets are separate from rules, so you can update them dynamically, which is faster than reloading the ruleset. Internally, sets may use bitmap or hash representations to facilitate **O(1)** lookup times.

ipset should be in your distribution's repositories. The syntax resembles **ip(1)** or **tc(1)**:

```
ipset create fail2ban hash:ip
ipset add fail2ban 1.2.3.4
```

Use bitmaps to store address or port ranges. Hashes are good for multiple disjoint values.

```
iptables -A INPUT -m set --match-set fail2ban src -j
DROP
```

Here, **set** is used to drop all packets whose source (**src**) address is in the **fail2ban** set. To block another host, just do **ipset add fail2ban 5.6.7.8**: no rule changes are needed.

FOSSpicks

Sparkling gems and new releases from the world of Free and Open Source Software

Our editor **Graham Morrison** is a fearless explorer of the internet – look, he's found some excellent Free Software on his travels!

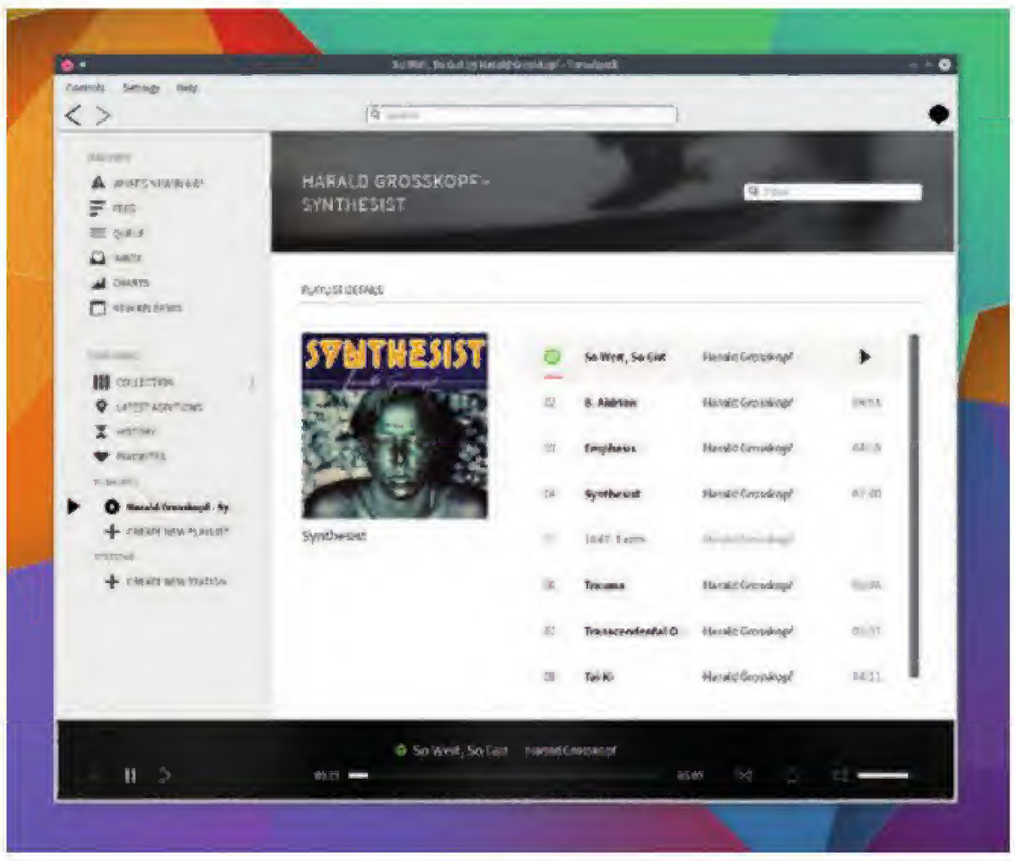
Music player

Tomahawk 0.8.99

From the minimalism of *XMMS* to the lyric-wielding, Wikipedia-reading *Amarok*, music players all fundamentally work the same way by playing your local music. Even if some add online music services or streaming internet radio, this is often as an afterthought. But not with *Tomahawk*. *Tomahawk* is trying to be a different kind of music player, and it's very much a product of our modern age of interconnected, music streaming social networks. Its main difference is that while it can (and does) scan your local music collection, it includes plugins to access over 20 online music sources, including Spotify, Google Play Music, Jamendo, SoundCloud, Ampache (OwnCloud), YouTube and Beats.

The Google Play Music plugin, for instance, operates as an expansion of your local music files. If you've synced your collection of Leonard Cohen's albums to Google Play (a service that's free for up to 50,000 songs), these will appear as your Google Play collection, and you can play and build playlists with these files and your local files.

What's more impressive is that if you search for an artist or a piece of music, *Tomahawk* will scrape through all of your configured sources, enabling you to construct playlists and albums from more than one source. *Tomahawk* doesn't make any distinction between sources, making it a fascinating way to access and discover new music. You can also share access to your music across the local LAN, or with your contacts through Google and Jabber.

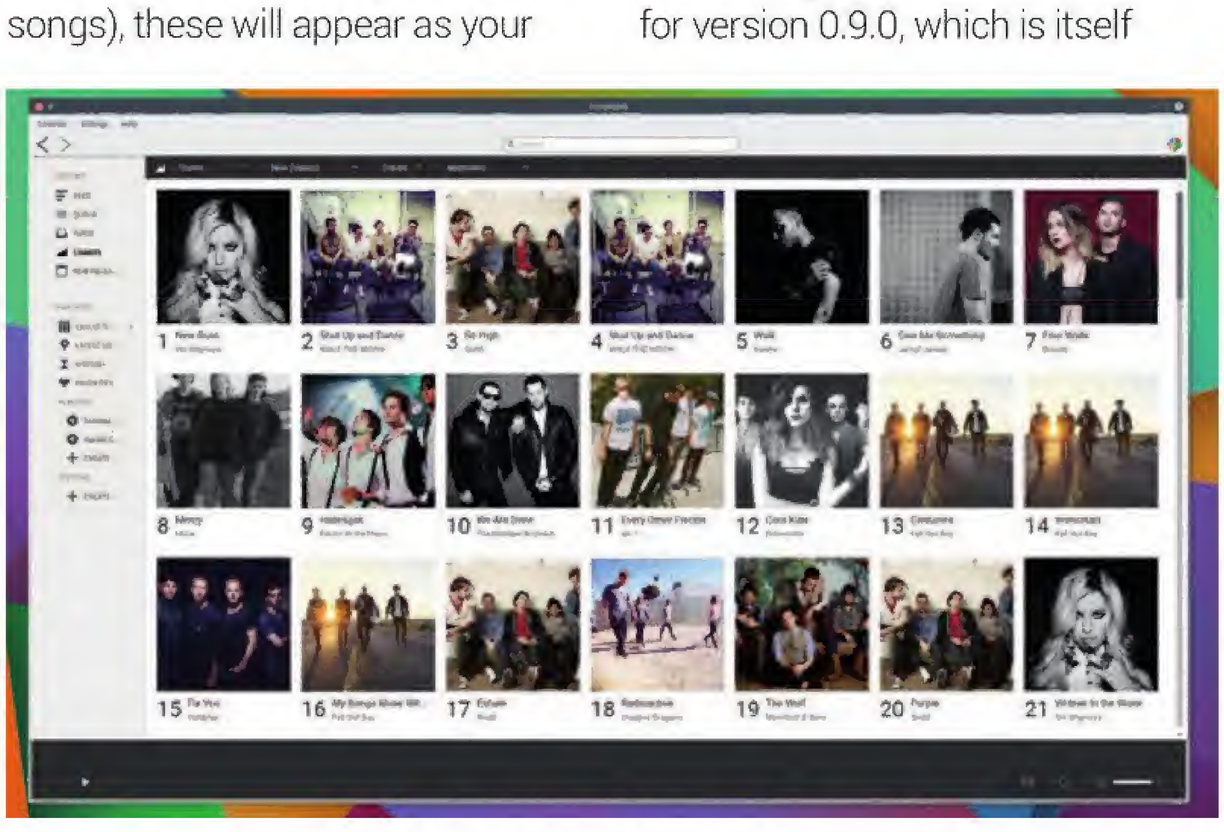


Interoperability, aggregation and cross-platform parity are the best reasons to use *Tomahawk*. There's even an Android version.

Groovy potential

The latest stable version was released in April (0.8.3). There are charts and new releases from multiple sources, including iTunes, Metacritic and Rovi. Clicking on a release will fill in the music sources from your enabled resolvers, so it won't always work. The version we're using is a release candidate for version 0.9.0, which is itself another major update. The developers have migrated to *Qt 5* from *Qt 4* and added more resolvers, including (an untested) one for Amazon Music.

There's no automatic import of your Spotify collection, but you can drag and drop playlists from the main Spotify application. We'd like more granular control over which sources are prioritised, and whether covers and live versions are returned from a search, as these options are only available when configuring a source. We'd also like to see more aggressive caching of your pre-configured playlists and searches. But these are small points when the application itself is doing so many new things. If you enjoy music, you need to check it out.



You can view and play chart music from all over the world, thanks to iTunes. Here's the charts from New Zealand, for instance.

PROJECT WEBSITE
www.tomahawk-player.org

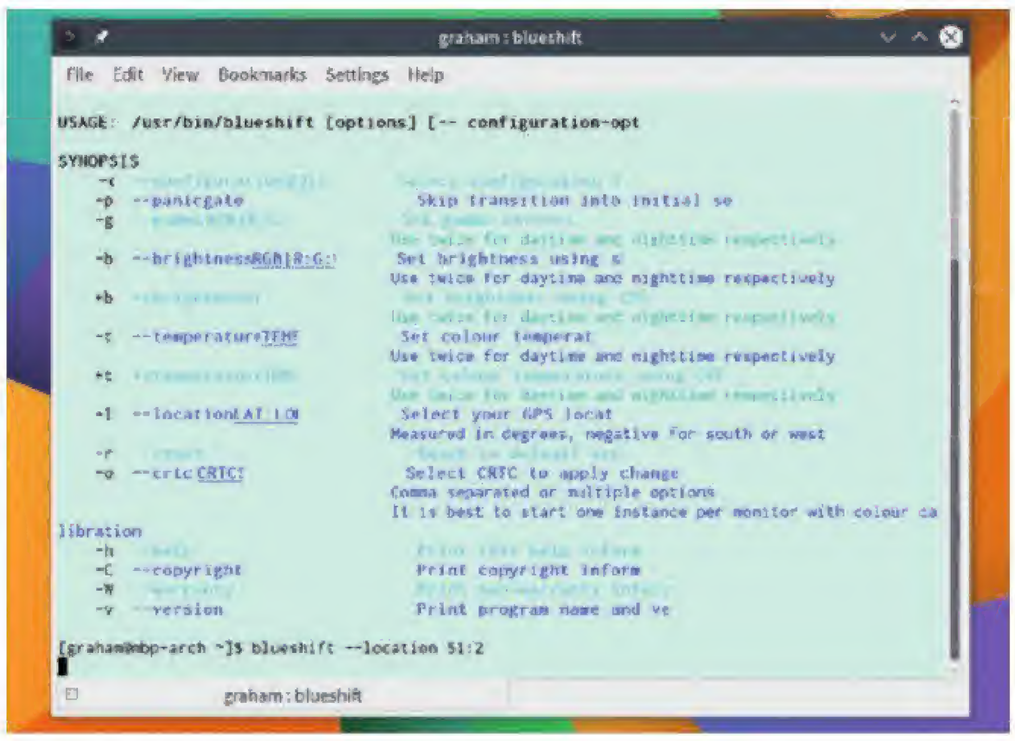
Screen colour adjuster

Blueshift 1.90.1

We published a brief tutorial on a wonderful utility called *Redshift* in issue 14. *Redshift* adjusts the colour temperature and intensity of your screen to better reflect the diminishing light of an evening or the darkness of night. This makes your screen easier on your eyes and your brain as there's less contrast and the hues are closer to dimmed light than the Mercury-vapour lamp white of the typical display. It helps many of us nocturnal workers get a better night's sleep.

It's such a useful tool that there are many forks and alternatives, and *Blueshift* is our favourite. Unlike a couple of alternatives that attach a GUI configuration panel to *Redshift* (*redshift-gtk* being one), *Blueshift* is driven purely from the command line. It takes many of the same arguments as *Redshift*: - you can set your location, for example, as well as the specific colour temperature you need from the screen. You can also push hues into the blue frequency range, as hinted at by *Blueshift*'s name, but we're not sure that this encourages concentration (as implied by the developers).

Blueshift is also a super-powered version of *Redshift*, and that power comes from its configuration files, which give you complete control over how colour transitions are handled. One example configuration is 'bedtime', for instance. This adjusts the colour temperature not by the light outside but by the time you want to head off to bed. Another example makes the light curves logarithmic, rather than linear. The 'xmonad' config file uses this window manager to map different light curves to different workspaces, and there's a file for reversing the colour palette when you're running low on battery power – excellent if you're running the screen on minimum brightness.



The only feature we miss from the original *Redshift* is the option to get your location from your IP address.

"Blueshift helps us nocturnal workers get a good night's sleep."

PROJECT WEBSITE
https://github.com/maandree/blueshift

Digital darkroom

LightZone 4.1.0

Linux has quietly become a photographer's dream studio. We now have more photo post-processing and management applications than ever, and they're nearly all brilliant. *LightZone* is one of these; so too are *Darktable*, *RawTherapee* and *AfterShot Pro*.

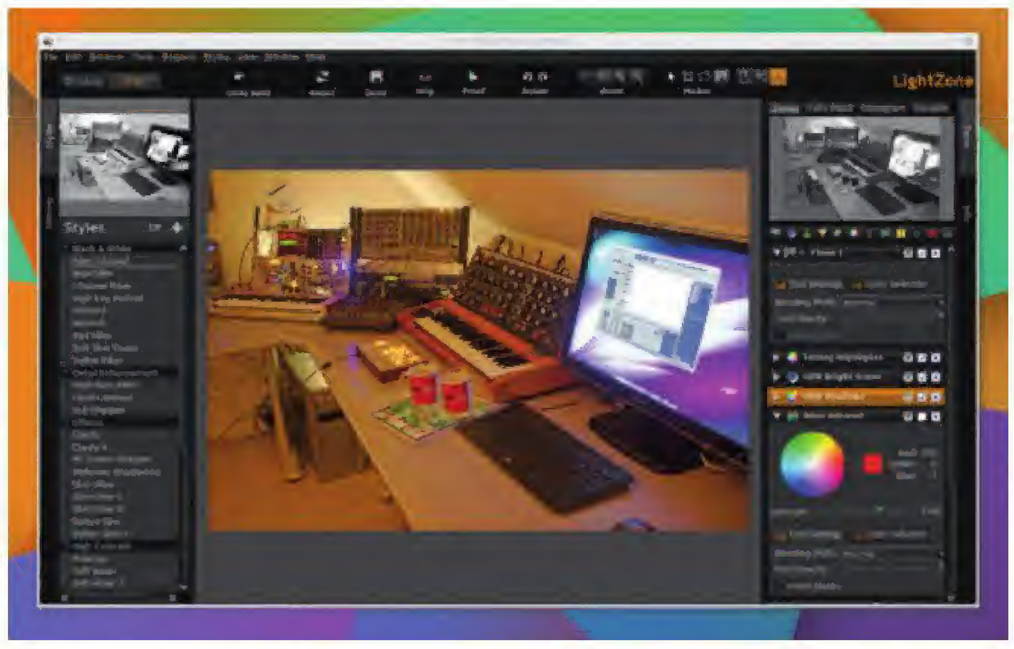
Since March last year, *LightZone* users have been asked to register to be able download from the main site. This is presumably to better track users, and over 80,000 have registered. But as the application remains open source, most of us Linux users will simply need to download the latest major update from our package managers.

There are two main modes – browse and edit, and even when dealing with the large RAW files produced by our DSLR, *LightZone* was much quicker than its competitors for both thumbnail rendering and editing.

Rather than an editing process based on Adobe's *Lightroom*, *LightZone* has a large list of 'Styles' that can be added to process an image, along with the regular sharpen, blur and colour balance effects you'd expect. RAW photos start off with exposure, noise and tint controls, and any further styles or processing you add can be moved up and down through the processing order.

Intuitive

Hover over a style and you'll see a preview of what the processes will do – perfect for choosing one of the nine black-and-white styles, for example. There's even red-eye removal and a cloning tool, which are rare in a post-processing environment. The clone tool is particularly good because unlike brushed cloning in something like *Gimp*, in *LightZone* you create a blended zone that acts like a portal between the source and the destination, and it can be moved after adding it to your edit queue.



LightZone is another great graphical application that was once proprietary and is now a fully fledged open source project.

PROJECT WEBSITE
http://lightzoneproject.org

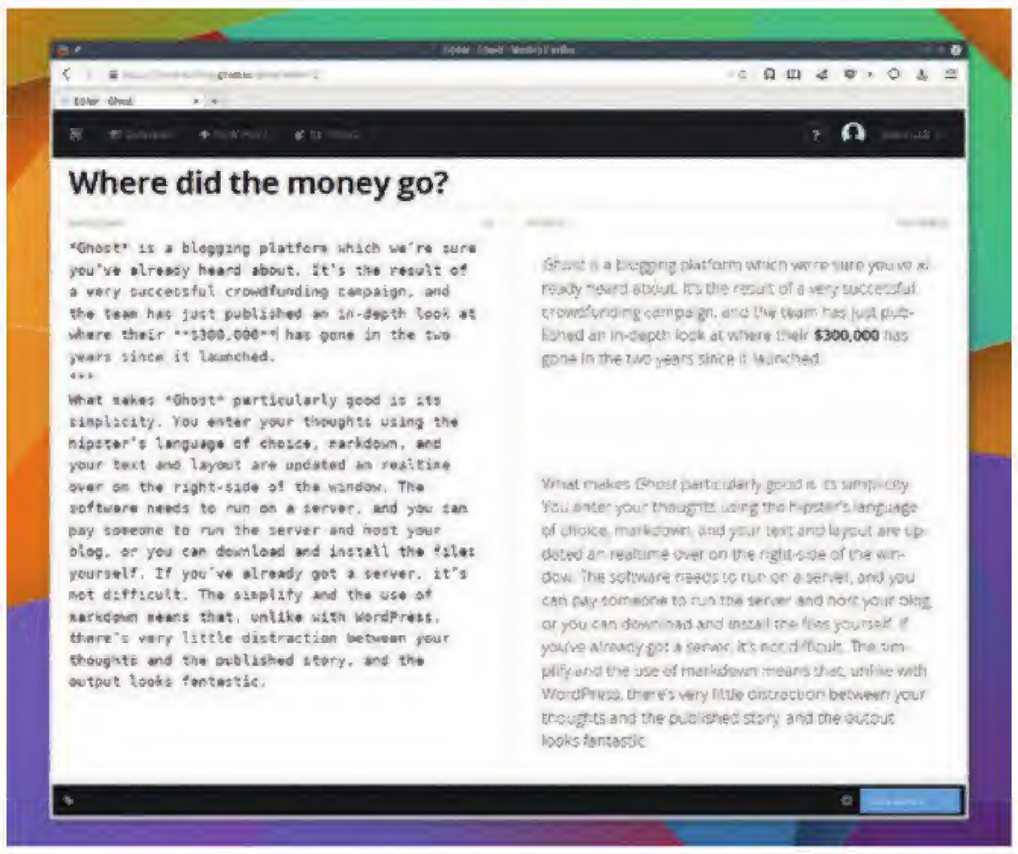
Blogging platform

Ghost 0.6.4

Ghost is a blogging platform that we're sure you've already heard about. It's the result of a crowdfunding campaign, and the team have just published an in-depth look at where their $300,000 has gone in the two years since it launched.

What makes *Ghost* great is its simplicity. You enter your thoughts using the hipster's language of choice, Markdown, and your text and layout are updated in real time over on the right-hand side of the window. Paragraphs, line breaks, emphasis, links, code and images are all handled quickly and easily, without any of the open-and-close-element hassle you get with HTML. The simplicity and the use of markdown means that there's very little distraction between your thoughts and the published story. And the output looks fantastic. By default, the same minimalism is carried forward into the published page, but *Ghost* is now well enough established that there are hundreds of themes available.

The software needs to run on a server, and you can pay someone to run the server and host your blog, or you can download and install the files yourself. The latest small update fixes more issues with the major 0.6.0 update, which appeared in mid-April. This included some functionality that had held us back from committing more time to *Ghost*, such as a spellchecker, mobile uploads and code injection, plus lots of new API hooks that will help developers without diminishing the overall user-experience, which is what's most important with *Ghost*. If you're looking at writing a blog, we'd highly recommend you take a look at *Ghost* first.



The *Ghost* blogging platform is already two years old, and it's come a long way from those early versions with little support.

"What makes Ghost a great blogging engine is its simplicity."

PROJECT WEBSITE
https://ghost.org

Arch package manager

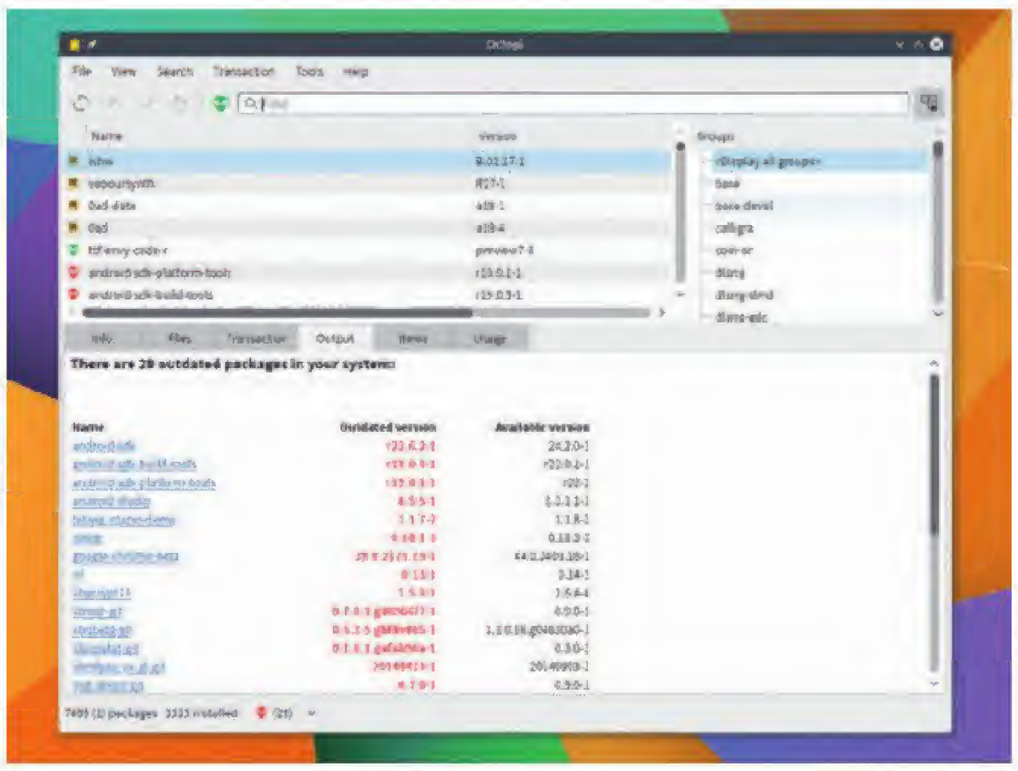
Octopi 0.7.0

In Arch distro land, we're quite happy using the command line for most day-to-day uses of *Pacman* (Arch's package manager) and *Yaourt* (its accompanying package manager for the user repository). But sometimes we long for a decent graphical interface to all those results and dependencies.

This is perhaps why there are several graphical interfaces to Arch package management, despite its hacker credentials, and *Octopi* is the latest. The best thing about *Octopi* is that it works everywhere. It works well on KDE 3, 4 and 5 desktops, *LXDE*, *Mate*, *Trinity* and *Xfce*, and across Arch-based distributions like ArchBang, Chakra, KaOS and Manjaro. Built on *Qt 5*, it's also very frugal with resources and we found database updates and refreshes as fast as the command line.

The main *Octopi* window is split into three: one panel lists packages; another Arch's metagroups; while the final panel is a tabbed view that switches between specific package information, the files it installs, and what happens when you do.

An essential addendum here is the latest news from Arch itself, which is a prerequisite before any distro upgrade you might perform. There's also a page on general application usage. Through the main window, there's a neat series of icons that quickly inform you if a package is part of the main repository or you've installed it yourself, and whether those packages need updating. Outdated packages get their own lists and a numbered reminder so you don't leave it too long before performing an update. We like the visual style very much. The small alien icons aren't over the top, and everything is beautifully functional.



Octopi also includes a notifier that can be set to sync the database at an automatic interval for updates.

"Sometimes we long for a graphical interface for our Arch packages."

PROJECT WEBSITE
https://octopiproject.wordpress.com

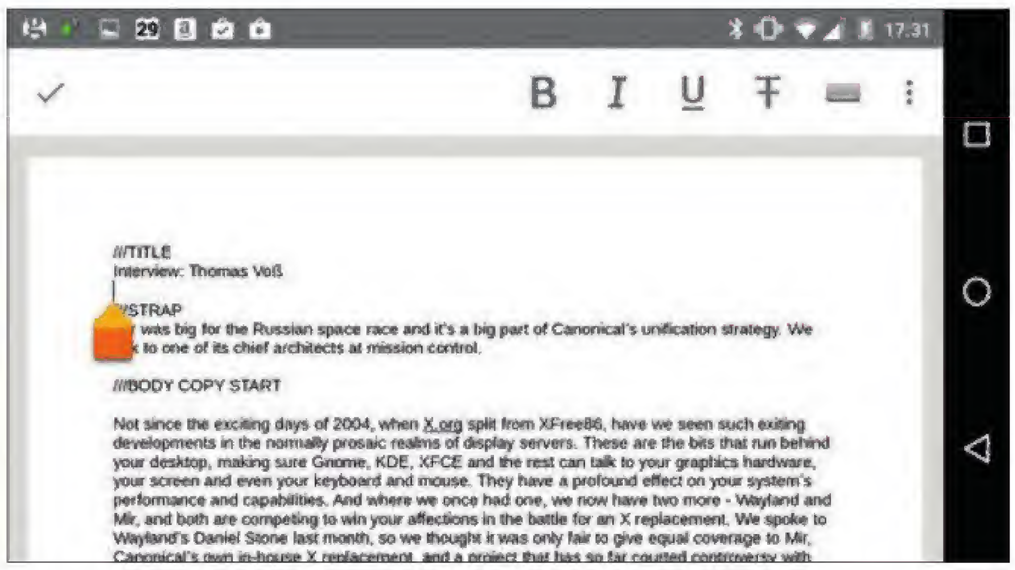
Office for Android

LibreOffice Viewer for Android 5 alpha 1

LibreOffice Viewer is published by the same people behind *LibreOffice* – The Document Foundation – and it's an important part of its strategy for getting open standards into the hands of as many people as possible, as well as keeping up with similar offerings from proprietary vendors. It's available through Google Play and as an APK that you can download and install yourself. It should also be available from the open source F-Droid repository by the time you read this.

It loads and views all the Open Document formats (.odt, .ods and .odp) as well as Microsoft's .docx, .xlsx and .pptx formats, and it's a great reader for files you might keep on your phone. We tested it with our own selection of documents and found the viewer to be just as good as the desktop version, rendering even complex documents without difficulty. Performance on our Nexus 5 was good, enabling us to zoom around pages and skip through slides with very little delay.

But the new and experimental feature that makes this release significant is the ability to edit documents. You need to enable this first in the settings panel, and after doing so, a cursor appears on what was previously a passive viewer. You can then move the cursor, type and make changes with the toolbar, just as you would on the desktop. It does need some refinement, but it works, and we can't wait to see this feature become more stable. This development is thanks to the work being done by Collabora, and one of the side-effects of creating a rendering engine that edits, works without *X.org*, and scales for Android devices is that it's leading to the development of the browser version, which should also be available later this year.



LibreOffice Viewer is now more than a simple viewer – with the new version, you can finally edit documents.

"You can move the cursor, type and make changes with the toolbar."

PROJECT WEBSITE
www.libreoffice.org

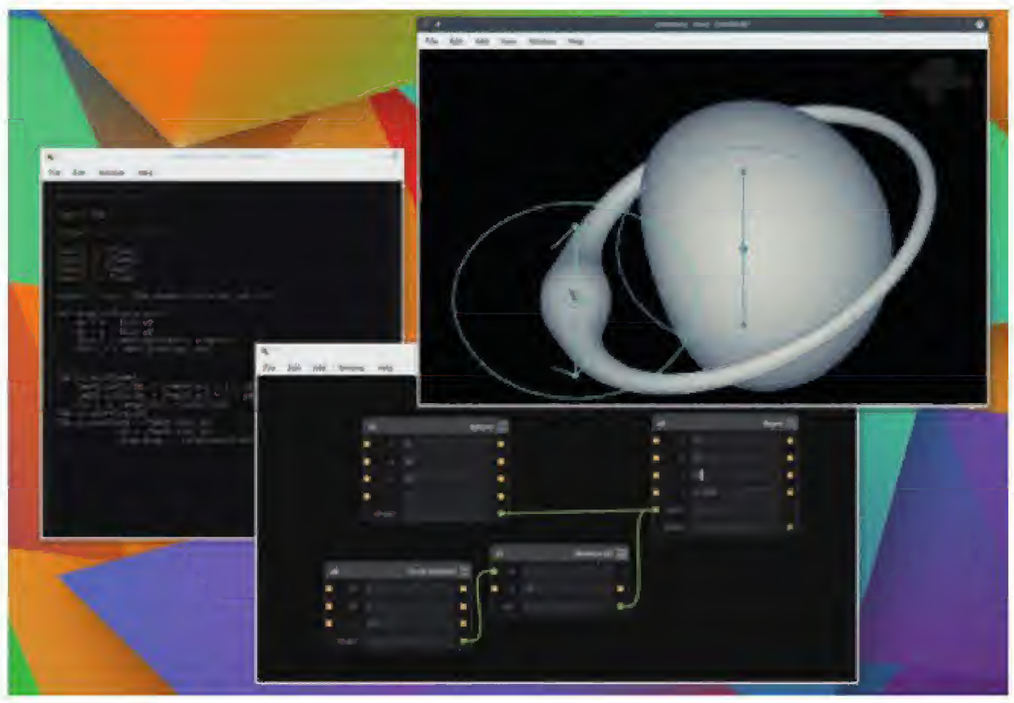
CAD modeller

Antimony (Git revision)

Antimony is a tool for Computer Aided Design, but it's like nothing we've seen before. Instead of interactive point-and-click modelling using primitives, *Antimony* builds objects by connecting nodes in a graph while still letting you manipulate values with the mouse. It's a little like a using a modular synthesizer, only instead of constructing sounds you're constructing 3D models.

There are two main windows. The first is the graph, which is where you add your objects, change their parameters and link attributes together. The second is the 3D view, which can also be split into front, bottom, side, back views or any of these separately. You can zoom around this view and move objects around, changing the parameters back in the graph when you do so.

The great thing about applying this to Computer Aided Design is that you're forced to use mathematical models and constructs, which is exactly what you need if you're planning to build something. For example, to build a 3D ring you first add a 2D circle and then connect the output of this to a 'revolve' function from the Transforms menu. This rotates the circle around the X axis, effectively making the circle follow a 360 degree path. There's also a script interpreter, and you can open any node and start editing the code used to render that specific element using a slightly augmented version of Python 3. Even without specific CAD knowledge, *Antimony* is a lot of fun. But if you're a mathematically minded designer, this kind of package must be incredibly powerful as the model output is algorithmically watertight and perfect for output that's going to be used to produce a physical model.



You can have as many windows and views open as you need, focusing on whichever parts of your model you want them to.

PROJECT WEBSITE
www.mattkeeter.com/projects/antimony

Video editor

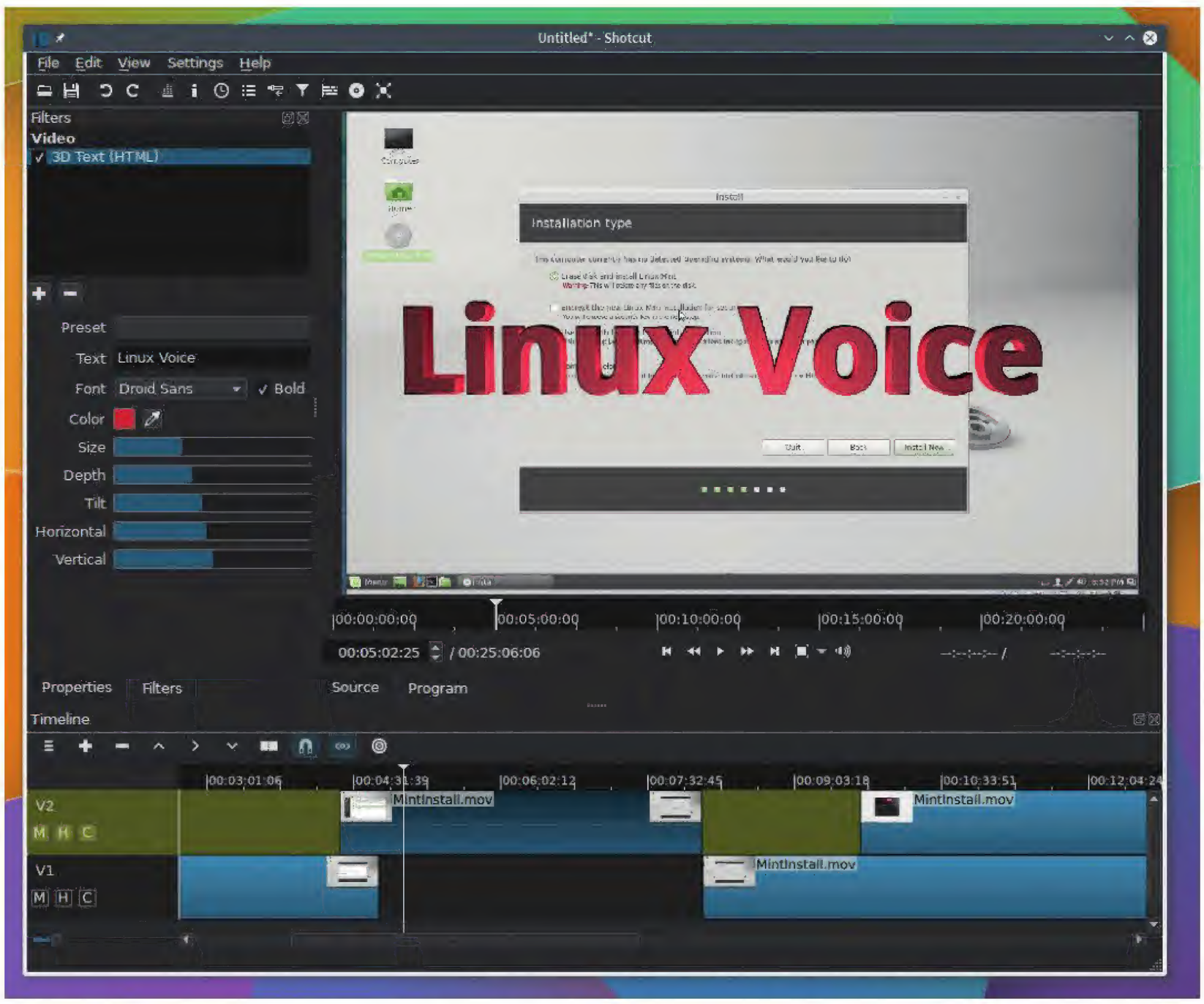
Shotcut 15.05

Despite there being some well established video editors on Linux, none have yet been able to make the process easy and intuitive for us. *Kino* was one of the most successful, however. It was powerful and capable of great results, but it was tricky to use. Unfortunately, *Kino* ceased to be developed in 2013. However, Dan Dennedy, *Kino*'s lead developer, has returned and created an all-new application, *Shotcut*, built atop his MLT framework. This means he doesn't need to completely re-invent the wheel when it comes to processing the video, and he can spend his effort working on how the user interface is going to work.

We really like *Shotcut*. The layout and workflow is supremely logical, and you can tell a lot of work has gone into its design. You import clips and add video tracks, split clips and crossfade between them. The edges of clips on the graphical timeline can be grabbed to extend them, and the cursor scrubs along the edit to make finding the right place to cut as simple as possible.

You can drag elements out of the background and into different areas, or leave them as floating windows, making it easy to create a layout that works for you.

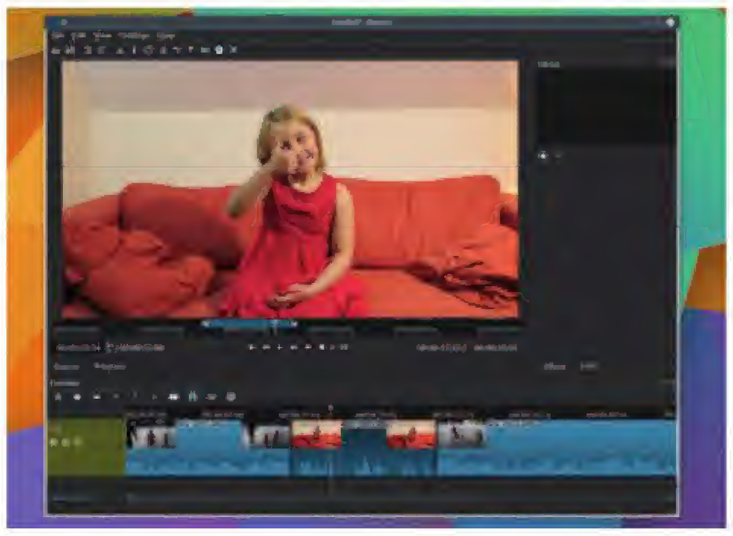
There's a small group of well implemented audio and video effects, and they give you all you need for most projects. There's colour balancing and grading, rotation, overlays, glow and opacity, for example, and creative effects include some excellent 3D text, sepia tones and waves. Best of all, a new beta feature allows you to farm out the processing of those effects to your GPU, which is the first time we've seen what's normally a professional addition in a piece of open source for Linux.



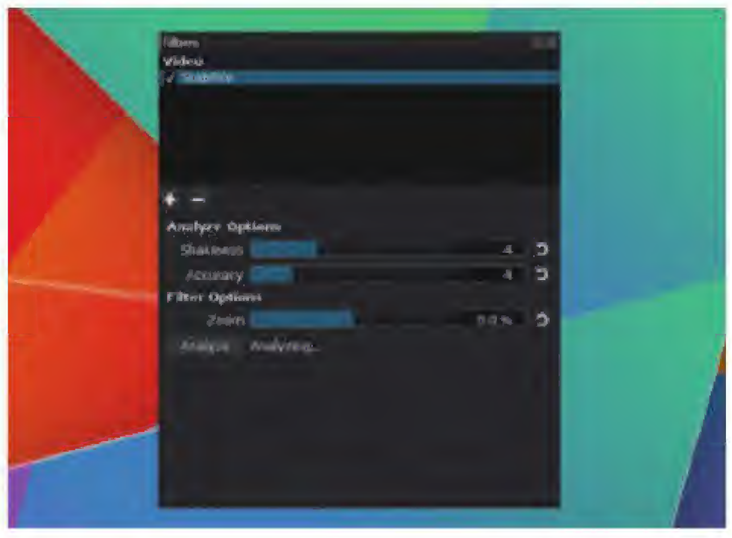
A feature unique to the Linux version is the ability to capture your screen and import directly into *Shotcut*.

PROJECT WEBSITE
www.shotcut.org

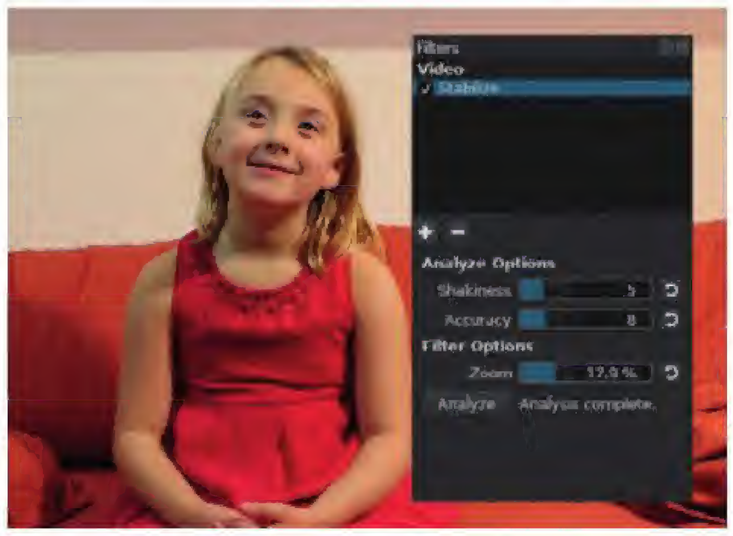
How it works: Image stabilisation



1 Limit the clip Image stabilisation takes its toll on processing and playback, so start by making your shaky clip as precise as possible.



2 Add the effect Use the video filters tab to add the stabilise effect and click on Analyse to let it work through the clip. It saves its output to a file.



3 Process the video When the status reads 'Analysis complete' you can adjust the options. Render the clip to a new clip to limit the CPU load

FOSSPICKS Brain Relaxers

Interactive fiction

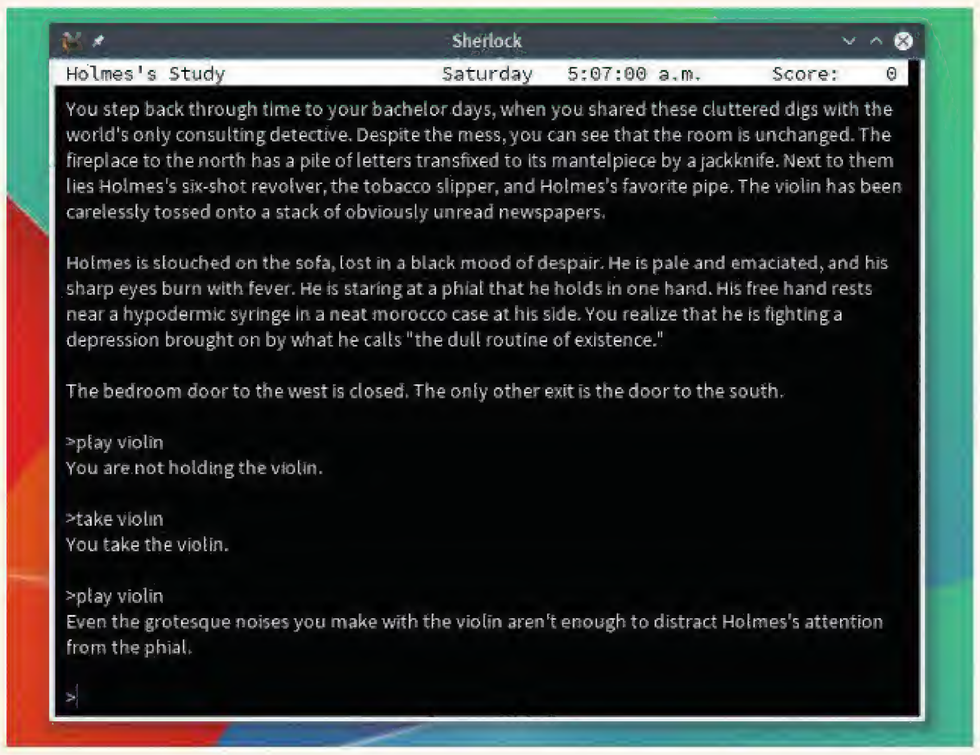
Fizmo 0.8.0 (b4)

Many years ago, there was a humble games genre where you could interact with a story by typing in commands. The computer might output "You are inside a building, a well house for a large spring. There are some keys on the ground here." And you might respond by typing, 'get keys'. By typing simple commands, moving through locations and solving riddles, you would become part of the story.

This genre was known as interactive fiction, and its most famous publisher was Infocom. In the process of creating games like the *Zork* series, *Hitchhiker's Guide to the Galaxy* (with Douglas Adams), or this writer's favourite, *Stationfall*, Infocom created a virtual machine-alike programming environment that's still used by interactive fiction 'aficionados' today. All you need to play these games is an interpreter.

There have been many interpreters over the years, and our current favourite is *Fizmo*. *Fizmo* is still being actively developed and version 0.8.0 is the first to take it away from the command line and into its own SDL window. It remains absolutely minimal yet supports nearly all of Infocom's games, including those with sound, and will load almost any of the brilliant games that are still being written and given away for free.

If you need some games, take a look at the interactive fiction database (**http://ifdb.tads.org**). This lists all known titles, and there's still huge interest in creating new ones. The authors of these will often make their work available completely free, and some are better even than those old classics. We recommend starting with Adam Cadre's brilliant *Photopia*.



Each year there's a competition to find the best new interactive fiction, which is still being written by fans today.

PROJECT WEBSITE
https://christoph-ender.de/fizmo

Role playing game

OpenMW

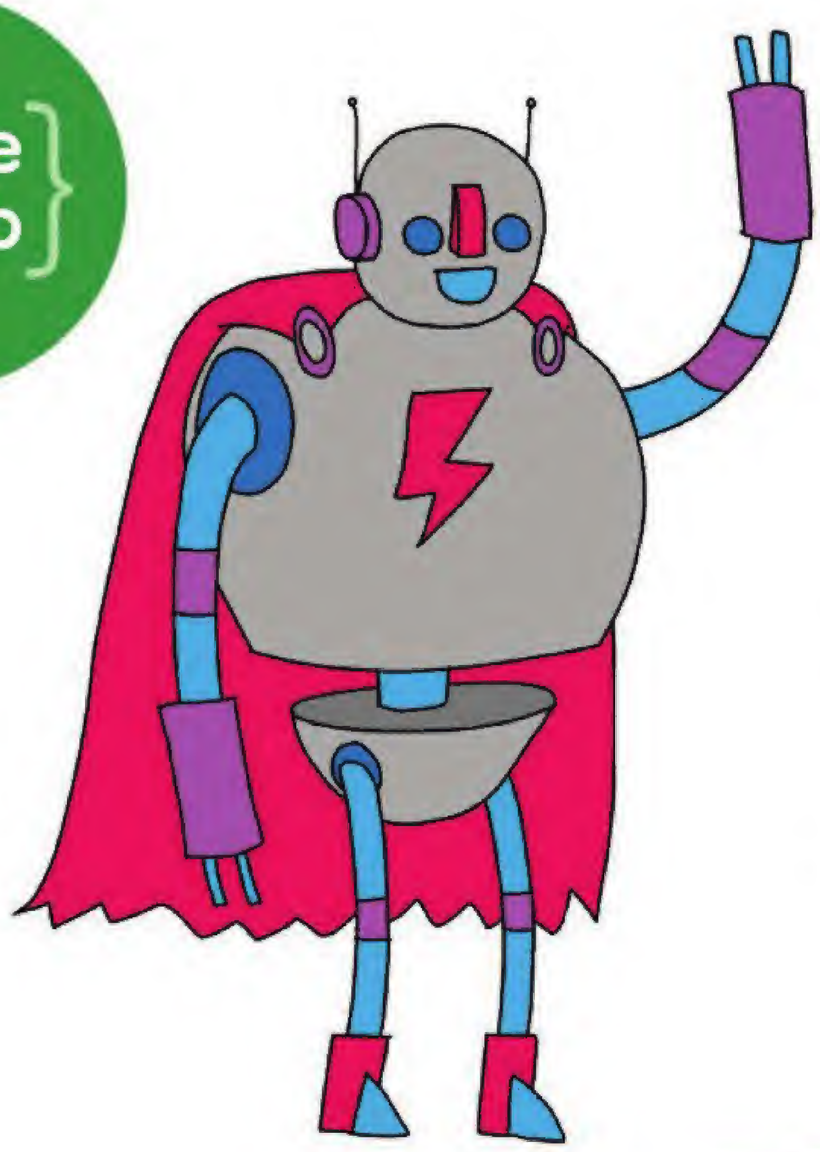
This isn't quite free software, because you need to source files from an original and still proprietary game. But *OpenMW* is worth the compromise because it's a complete recreation of the games engine behind one of the best ever PC role playing games – *Morrowind*. Released in 2002, *Morrowind* is the third in the *Elder Scrolls* series, coming after *Daggerfall* (see our tutorial on p88 getting this to run for free) and before *Oblivion*. It consists of a huge open world viewed in first person, and while this is a recreation of that original game, *OpenMW* has ambitions beyond straightforward recreation. *OpenMW* includes an editor, for example, enabling players to modify and even create their own environments and adventures. For *Morrowind* players, this is brilliant because there's still a huge community playing the game and making modifications that improve nearly every aspect of the original.

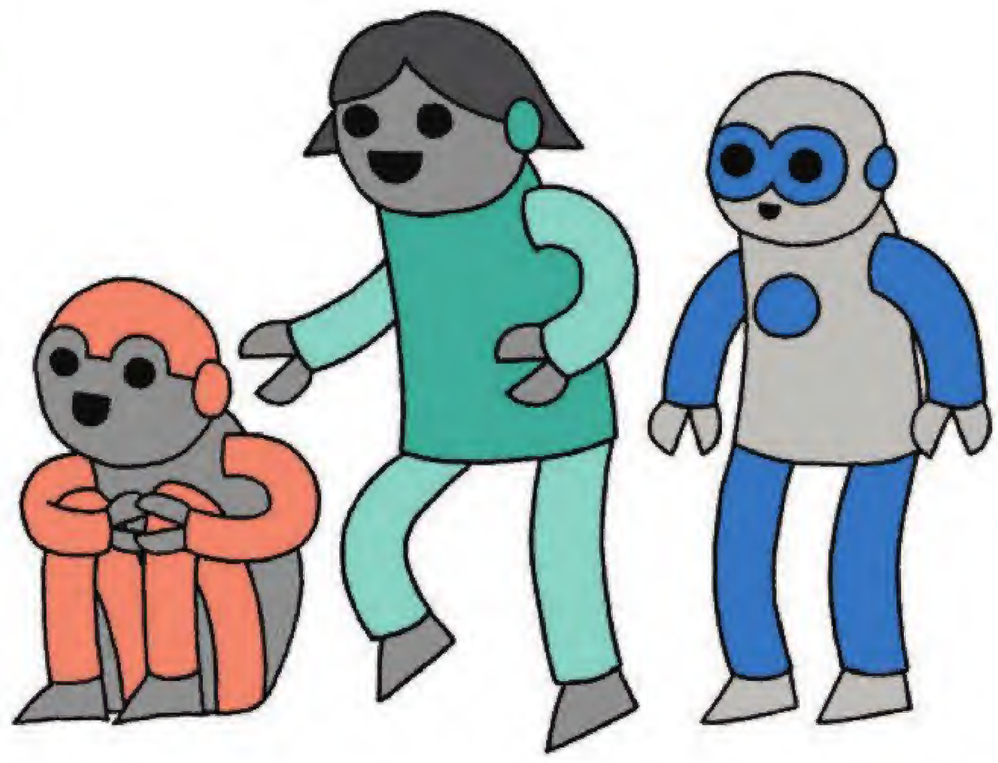
OpenMW is also easy to use. You start with the launcher, which will ask for the location of the installed files or the mounted drives, and it sucks up data from *Morrowind* itself and both of its expansion discs, if you have them. A configuration panel is then used to generate display settings with OpenGL for graphics rendering and for launching the game, with or without expansions and mods. The quality of the recreated engine is staggering, and while it's a bit of a CPU and GPU hog, this will hopefully improve when the migration to *OpenSceneGraph* is complete.

Before *OpenMW* will work, you'll need the media assets and game data from an original copy of *Morrowind*.

"The quality of the recreated game engine is staggering."

PROJECT WEBSITE
www.openmw.org





Can you help inspire the next generation of coders?

Code Club is a nationwide network of volunteer-led after school clubs for children aged 9-11.

We're always looking for people with coding skills to volunteer to run a club at their local primary school, library or community centre for an hour a week.

You can team up with colleagues, a teacher will be there to support you and we provide all the materials you'll need to help get children excited about digital making.

There are loads of ways to get involved!

So to find out more, join us at **www.codeclub.org.uk**

LINUX VOICE

TUTORIALS

Dip your toe into a pool full of Linux knowledge with eight tutorials lovingly crafted to expand your Linux consciousness

Ben Everard
Is making a DOS game which runs batch jobs on Ubuntu Core using Bluetooth and Android.

I recently attended a talk by Cory Doctorow, information freedom campaigner and science fiction author (just before doing this month's interview). One of the things he spoke of was the importance of pragmatism in the fight for digital freedom.

No one is perfect. It's impossible to avoid every piece of proprietary software. Even if you use a fully free distro (such as Trisquel), there's still proprietary microcode running on your CPU. There's still proprietary firmware in your hard drive (and most likely on other parts of your machine). Even if you somehow managed to overwrite this firmware, the hardware is still closed. Likewise, you almost certainly use some data harvesting web service such as Google search or Facebook.

The only way to be digitally free is to abandon the digital world altogether, and although that may seem like an attractive option at times, it doesn't really benefit anyone. A far better solution is to recognise that you inevitably support some organisations with bad practices and try to counterbalance that by supporting organisations with good practices.

You can cleanse yourself of digital guilt by donating to ORG, FSF, EFF, SFLC or one of the other organisations that campaigns for digital rights.

ben@linuxvoice.com

In this issue...

LINUX VOICE TUTORIAL

KDE CONNECT: GET DESKTOP NOTIFICATIONS FROM A PHONE

GRAHAM MORRISON

Share files, check battery status, read notifications and add remote control to and from your phone.

WHY DO THIS?

- Check phone status from your desktop
- Upload and download files
- Share clipboards

Laptops and phones have become inseparable. They can often be found huddled together on desks and breakfast bars across the land. But it's only now that tools are being created to better unify them. Canonical is trying its best with Ubuntu Phones, and both Apple's OS X and Google's Chrome OS are starting to blend their desktop operating systems with their mobile ones.

Linux has had these kinds of tools for some time, and our favourite is *KDE Connect*. *KDE Connect* sends all kinds of useful information about your Android phone to your desktop, including notifications, messages and files, and lets you remote control your desktop or use your phone's keyboard for input. Yes, *KDE Connect* does work best with KDE – it's still primarily a KDE 4 application, although we installed it in Plasma 5 alongside the new widget and it worked well. But an additional install called *KDE Connect Indicator* adds much of the same magic to almost all other desktops too, so everyone can get metaphorically closer to (or literally further away from) their phones.

Step by step: Link your phone and your desktop

1 Installation

Most distributions will have a package for *KDE Connect* and installation should be simple. However we found the Ubuntu/Kubuntu package a little old, so we'd suggest using the following PPA:

https://code.launchpad.net/~vikoadi/+archive/ubuntu/ppa/

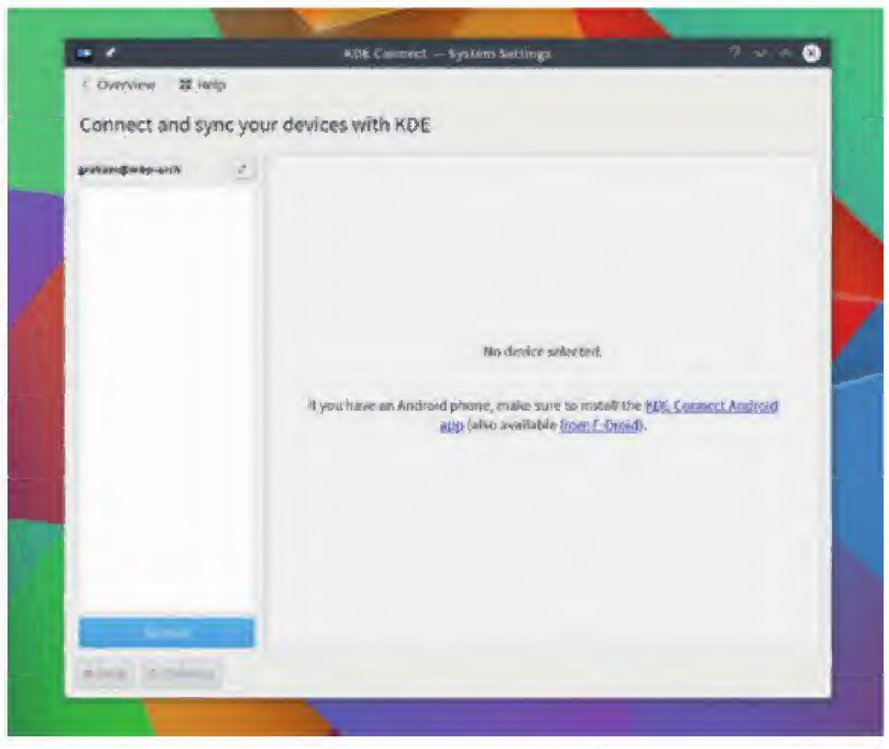
If you've not used a PPA before, simply follow the instructions to add the required repositories to your system and you'll find *KDE Connect* is now available. On Arch, we installed **kdeconnect-git** from the user repository because it was a much more up-to-date version, and it didn't require any weird dependencies.

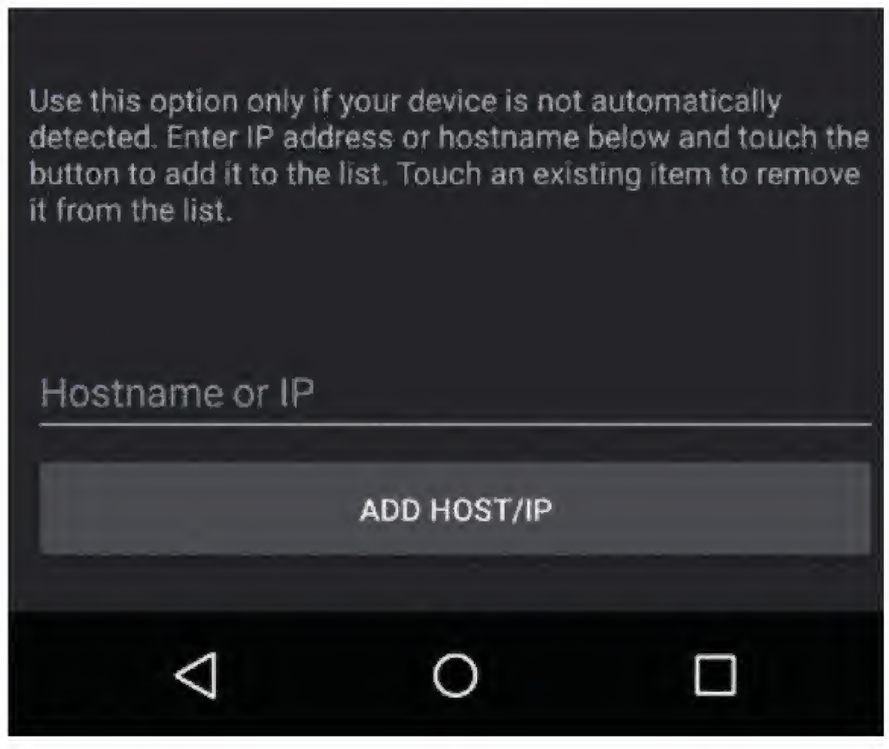
You'll need to restart the KDE desktop after the installation because the tool itself adds a settings panel and runs a background daemon that's going to wait for incoming connections. If you open the settings panel now from the System Settings menu, no devices will be detected and you'll be informed that you need to install the app on your phone...

2 Get and configure the app

... There are two sources for the Android 'KDE Connect' app: the Google Play store and the F-Droid open source package manager. The app is a free download on both (and open source too) and at the time of writing they both offered the same version for download – version 0.8g.

After the app has installed and you've launched it, there's a good chance your laptop/desktop will appear in the list of available devices, as detected by KDE's auto-discovery. If not, you should click on the menu icon in the top-right and select 'Add Devices By IP'. You can then use the button at the bottom of the screen to add and enter either an IP address for your computer, or a hostname if it resolves across your network. We found that after doing it once, our computer was always detected, even without adding the IP address manually. Type **ifconfig -a** or **ip addr** on your KDE machine's command line to get its IP address.





3 Make the connection

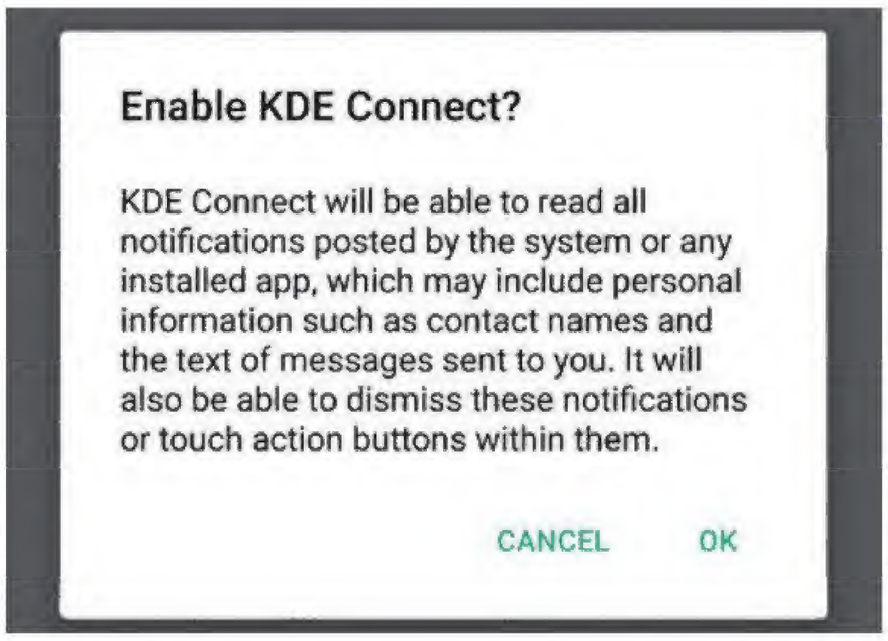
At the same time that your desktop appears on your phone, you should find that KDE's settings panel now lists your phone/Android device too. You now need to pair the two devices together, which you can do from your Android device or your computer running the KDE desktop. On your Android device, select your KDE machine and after it says 'Device Not Paired' press the 'Request Pairing' button. You should immediately get a notification on your KDE desktop to say there's a pairing request from your Android device and you need to click on 'Accept'.

If you miss this, just press the back arrow on your Android and try again. As soon as you've accepted the request, your Android device will now list your desktop beneath 'Connected Devices' rather than beneath 'Available Devices'. Back at the KDE settings panel, your device will have turned green to show it's authenticated and you'll no longer be able to choose the 'Pair Device' option.

4 App configuration

Back on the Android device, select the desktop machine and you'll see a few options. The first opens a simple set of transport controls so you can play, pause, skip and change the volume of media being played back from your computer – ideal if you've got your laptop plugged into a television. The second is labelled 'ping'; pressing this will open a simple notification window on the desktop.

The last button in the list will turn your device's touchscreen into a touchpad for your desktop, and it works rather well. Moving the cursor is very sensitive and capable of offering good control. You can use the touchpad to turn a single tap into a left-click, a two-finger tap into a right-click and a three-finger tap into the middle button. These options are also available from the app's menu.



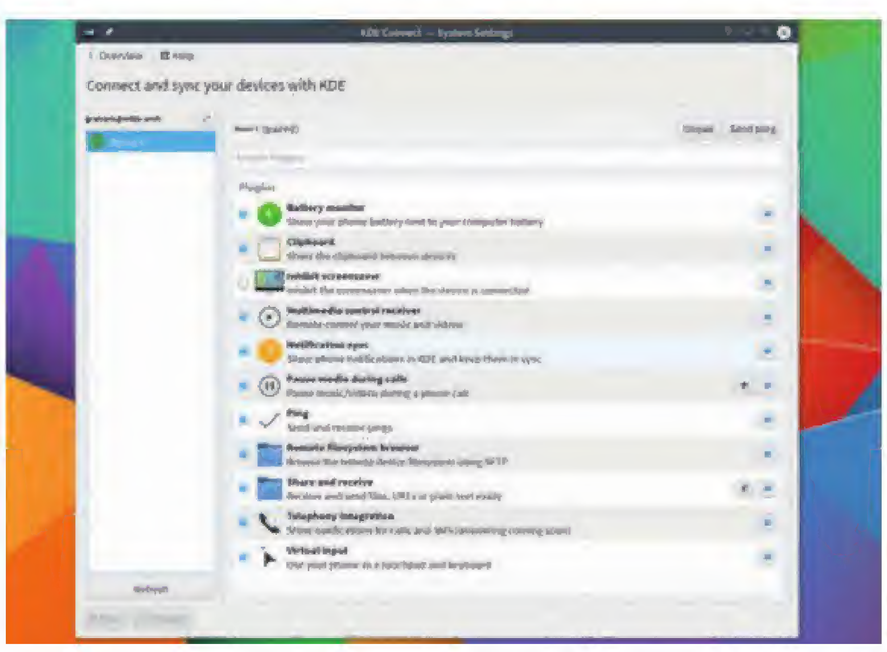
5 KDE configuration

KDE gives you specific control over which parts of the app you want imposing themselves on your desktop, and these options are available from the now-populated KDE settings panel for *KDE Connect*. Top of the list is the ability to watch your Android device's battery drain from the comfort of your desktop, just in case you weren't paranoid about it enough. The second option is particularly awesome, as it shares the clipboard contents between the two devices. Select a URL on KDE, for example, and you can simply paste the same string from your phone. It works like magic! You can also choose to enable notifications, ping or the multimedia controls, as well as whether incoming phone calls pause your music playback. This works with almost any media player.

6 Using KDE Connect

There's another part of the desktop, and that's the widget that displays device-specific notifications/interactions and the remaining battery life of your device, as well as giving quick access to the settings panel and the file sharing capabilities. For Plasma 5, you can install this just as you would any other widget, using the 'Add Widgets' menu from the background.

You should now be able to share (for example) a URL on your phone and see it open the default application on your desktop – usually a file browser. The folder icon on the widget will also open an SFTP connection to your phone, so you can browse its filesystem from your KDE desktop – we had to add **/storage** to the end of the path as there seemed to be a permissions problem with Android. LV



LINUX VOICE TUTORIAL

WIIMOTE-TRIGGERED SELFIE MACHINE

LES POUNDER

Les Pounder digs out his neglected old Nintendo Wii and hacks together the latest in selfie technology.

WHY DO THIS?

- Learn Python
- Use new types of inputs
- Repurpose old technology

TOOLS REQUIRED

- A Raspberry Pi Model Pi 2 or B+
- Raspbian operating system
- PiCamera
- Nintendo Wiimote
- A Bluetooth dongle
- Monitor, keyboard, mouse and power supply for the Pi

The Nintendo Wii games console was released in November 2006 to much fanfare due to a novel method of input – it used a candybar shaped controller with a number of sensors such as an accelerometer, and an IR (infrared) sensor, which when used with the included sensor bar could locate your position and use it to control your character. The controller (which became known as the Wiimote) also featured a vibration motor for haptic feedback such as gunfire. But how did the Wiimote connect to the Wii? Well it used good old Bluetooth to provide two-way communication between it and the console, and the aforementioned sensor bar was really a series of IR LEDs and a power supply. The Wiimote IR sensor would calculate its position relative to the LED and then communicate that to the console.

Sadly the Nintendo Wii has ceased production – but we're going to give its hardware a new lease of life, by building a Raspberry Pi-powered selfie machine to trigger taking a picture and recording a short video.

Setting up the camera

To install the Raspberry Pi camera module, your Raspberry Pi will need to be turned off, as the camera is a rather delicate piece of kit. Locate the CAMERA port on your board (between the HDMI and Ethernet ports). Gently pull the clip upwards to open the port ready for the camera cable. The camera cable will slide into the port, with the silver contacts facing the HDMI port and the blue band on the reverse facing the Ethernet port. Once the cable is slid into place, gently push the clip back into place to grip the camera cable into the port.

The Camera has its own dedicated port on the Raspberry Pi and fits in rather securely, but be careful as it's quite fragile.

Taking a picture with your Pi is really rather easy. You too can take high-quality cheesy selfies with ease!

With the camera hardware installed, attach your components and peripherals, then boot your Raspberry Pi to the desktop. With your Raspberry Pi connected to the internet, open a Terminal (the icon for which is located in the top-left of the screen and looks like a black computer monitor).

In the terminal window, type the next two lines; at the end of each line, press Enter:

```
sudo apt-get update
sudo apt-get upgrade
```

The first line updates the list of repositories, which are places that contain Raspbian software packages, to ensure that we have the latest software lists for reference. The second line instructs our Raspberry Pi to compare our installed software with that provided by the repositories and, if there are any upgrades, to download and install them. By completing this step we can confirm that we have downloaded the PiCamera Python package, which we will use later in this project.

We will now issue another command in the terminal to start the configuration tool:

```
sudo raspi-config
```

In the menu that appears, navigate to option 5, Enable Camera, using the arrow keys, and press Enter to confirm your entry. Choose Enable and press Enter, then navigate to Finish to exit the config tool. If you are prompted to reboot then do so and return to the Raspbian desktop to continue.

Now we need to test that the camera is working.

Taking a still image

Our first command is called **raspistill**, and as you may have guessed, it uses the camera to take a still image. To use the command we need to type the following into the terminal:

```
raspistill -o test.jpg
```

This will open the camera preview and you should see an image on screen. After around five seconds the camera will take your pic and save it as **test.jpg**. Once **raspistill** has completed it will return control of the terminal to you. If you open the File Manager, which can be found in the task bar, you'll see **test.jpg** in your home directory, which is where we ran the original **raspistill** command.

To test video recording, we can use **raspivid** with the following command in the terminal:

```
raspivid -o test.h264
```

Again this will launch the preview window but it is now recording video and will do so for the next 10 seconds. Once finished, the terminal control will be returned to you. To watch your video, type the following into the terminal:

```
omxplayer test.h264
```

With our camera tested, it's now time to set up Bluetooth.

Setting up Bluetooth and CWIID

For our project we will need a Bluetooth USB dongle – we used an ORICO Bluetooth 4.0 dongle from Amazon (**http://bit.ly/LV17Bluetooth**) as it had a decent range of 10 metres and consistently connected first time with our Raspberry Pi/Wiimote combo. Plug your Bluetooth dongle into a spare USB port, then open a new Terminal window and type in the following:

```
sudo apt-get install bluetooth
```

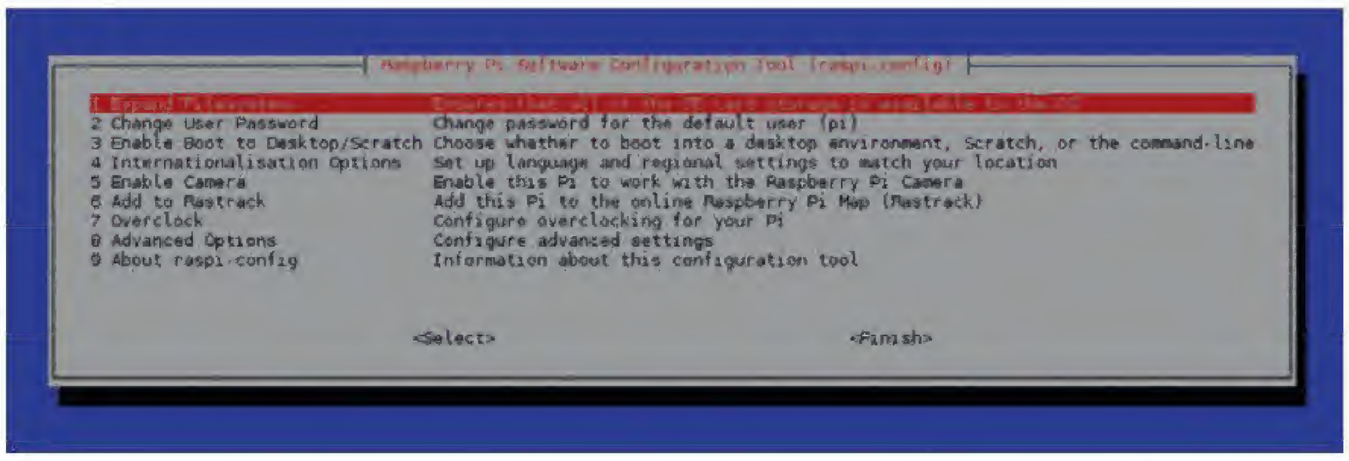
This command will install all of the dependencies for using Bluetooth with our dongle – it will also take some time, so perhaps pop off for a cup of tea and come back in a few minutes.

To enable Python to talk to the Wiimote, we need to install a library, and we do that using the following command in the terminal:

```
sudo apt-get install python-cwiid
```

CWIID, pronounced "seaweed", is a Python library that handles communication between your Raspberry Pi and the Wiimote. In this project we are using CWIID with Python, but there are also packages available in the repositories to enable your Wiimote to be used as a mouse/presentation device – see **https://help.ubuntu.com/community/CWiiD** for more details.

With Bluetooth and CWIID installed, it's time for us to move on to part 3 – putting it all together.



The *raspi-config* menu is a handy suite of tools to configure certain elements of your Raspberry Pi.

Building the selfie machine

In Parts 1 and 2 we have successfully set up our camera and Bluetooth dongle and now our focus shifts to creating the code that will control our selfie machine. Our project will be written in Python 2.7, this is due to CWIID not having a Python 3 library. So we start by opening the *Idle* text editor, which you can find in the main menu under 'Programming' and then under 'Python 2'. Once *Idle* is open, click on File > New to open a blank document. Straight away save this file as **selfie.py** by clicking on File > Save.

We start the code by importing the libraries that will form the basis of the project:

```
import cwiid
from time import sleep
from datetime import datetime
import picamera
import os
```

We first import the **cwiid** library that we installed earlier; we then import a function from two libraries. From the **time** library we import the **sleep** function, to control the speed of our project, and from **datetime** we import the **datetime** function, used to add a timestamp to our images. Our last two imports start with the **picamera** library, used to control the camera, and finally the **os** library, used to run Linux shell commands in Python.

Next, we create a variable called **button_delay** to ensure that input is read only once per press:

```
button_delay = 0.1
```

Raspberry Pi Camera

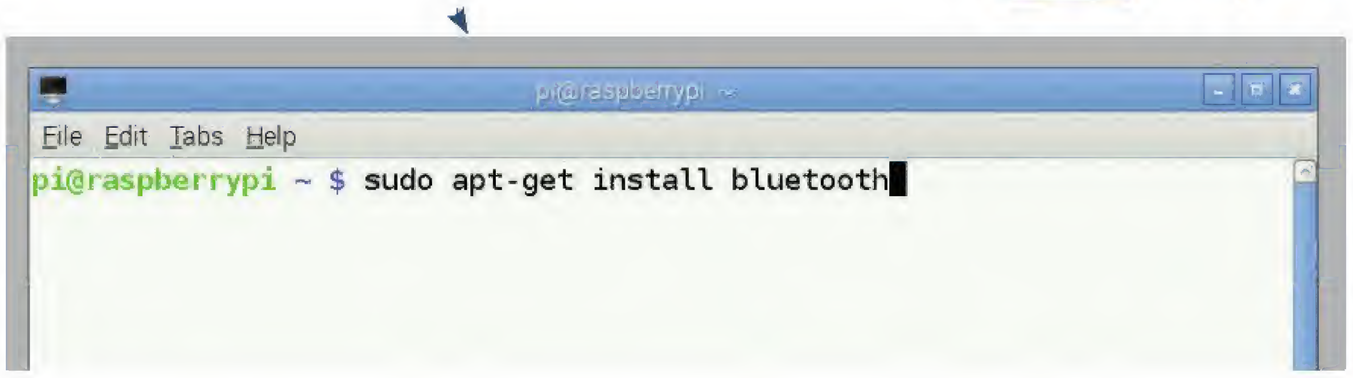
The Raspberry Pi Camera comes in two flavours. In this project we used the standard version that caters for all scenes, but there is also the Pi Noir camera, which is used in low light scenarios along with an infrared light source to record video and take pictures at night. It is commonly used in nature photography such as bird boxes, as in this tutorial from the Raspberry Pi Foundation: **www.raspberrypi.org/learning/infrared-bird-box**.

The Raspberry Pi Camera can also shoot at high speed using the **raspivid** command. Speeds of up to 90 fps are possible at 640x480 resolution, enabling you to easily create high-speed photography. You can try it by running

```
raspivid -w 640 -h 480 -fps 90 -t 10000 -o test90fps.h264
```

This will capture 10 seconds of video at 90 fps. When played back, the video will be running at a third of its normal speed, due to the slower 29.97fps of normal video playback. The video will look like slow motion but will capture sharp images at every frame.

If you would like to know more about high speed photography read **https://www.raspberrypi.org/new-camera-mode-released**, and for general camera information head over to Dave Jones' great documentation at **http://picamera.readthedocs.org/en/release-1.10**. No matter which camera you may choose, they both work with the PiCamera Python library in the same manner and they also work with the **raspistill** and **raspivid** commands.



Installing Bluetooth requires a lot of dependencies, but don't worry: the **apt** packaging tool will do all the hard work.

Our next section handles creating three functions, which will handle taking a picture, recording video and finally displaying the picture on the screen. We start with taking a picture.

Functions are defined, in that they are given a name that can be called, and when that is the case the contents of the function are run. Our first function is called **takepic**, and it also has the word **pic** in brackets. This is an argument, an extra option passed to the function when it is called. All the code indented underneath the **def takepic(pic):** line is part of the **takepick()** function:

```
def takepic(pic):
  with picamera.PiCamera() as camera:
    camera.start_preview()
    for i in range(5):
      wii.rumble = 1
      sleep(1)
      wii.rumble = 0
      sleep(1)
    camera.annotate_text = (pic)
    camera.capture((pic))
    camera.stop_preview()
```

Line 2 shortens the long **picamera.PiCamera()** into **camera**, which is much easier to work with:

```
with picamera.PiCamera() as camera:
```

We then indent once again and trigger the camera preview function, which will show a live shot from the camera:

```
camera.start_preview()
```

Our next section of code is still inside the function and makes the Wiimote's motor vibrate five times, giving the user a countdown until the photo is taken. We use a **for** loop to iterate five times, turning on the motor for 1 second, then turning it off for 1 second:

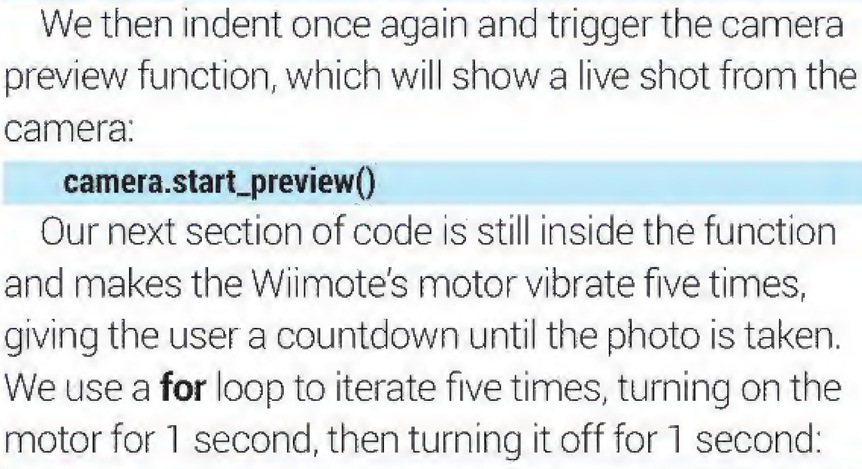
```
for i in range(5):
  wii.rumble = 1
  sleep(1)
  wii.rumble = 0
  sleep(1)
```

Our next line of code is not part of the **for** loop, but is still inside the **with** conditional that we created at the start of this function. We can add the time and date that the picture was taken as text on the image:

```
camera.annotate_text = (pic)
```

Our last two lines of code handle capturing the picture and saving it as the filename contained in the pic variable. Finally we close the preview window:

```
camera.capture((pic))
camera.stop_preview()
```



Running the code will produce a series of outputs to the shell; these are instructions to the user.

Recording video

Our second function controls the recording of a short video, and the structure of this function is very similar to that of the previous function.

We start with naming the function, creating an argument named **vid** which will later contain the timestamp for the video. We then also repeat the shortening of the **picamera** function:

```
def takevid(vid):
  with picamera.PiCamera() as camera:
```

Indented into the **with** statement, we have the next two lines of code: the first sets the video resolution to the HD 720p, which give us the best compromise between video quality and small filesize. We then start recording the video, passing the **vid** argument that we will later create, and use string concatenation to attach **'.h264'**, the video format which is used to record the video, to the timestamp (**vid**):

```
    camera.resolution = (1280, 720)
    camera.start_recording((vid)+'.h264')
```

We next add the timestamp to the video in the same manner as we did for the previous function. Then we start the preview window to help the user frame their shot:

```
    camera.annotate_text = (vid)
    camera.start_preview()
```

We now create a **for** loop that will iterate 10 times to rumble the Wiimote 10 times; you'll see on the last line of this snippet that we have added **camera.wait_recording(1)** this is a unique function for recording video and is used in place of the **sleep** function. Using the **wait_recording** function the program will check to ensure that there is enough disk space for your video:

```
    for i in range(10):
      wii.rumble = 1
      sleep(1)
      wii.rumble = 0
      camera.wait_recording(1)
```

Finally we stop the preview window and recording:

```
    camera.stop_preview()
    camera.stop_recording()
```

Our final function handles showing the user the last picture that was taken; we name it **showpic**:

```
def showpic():
```

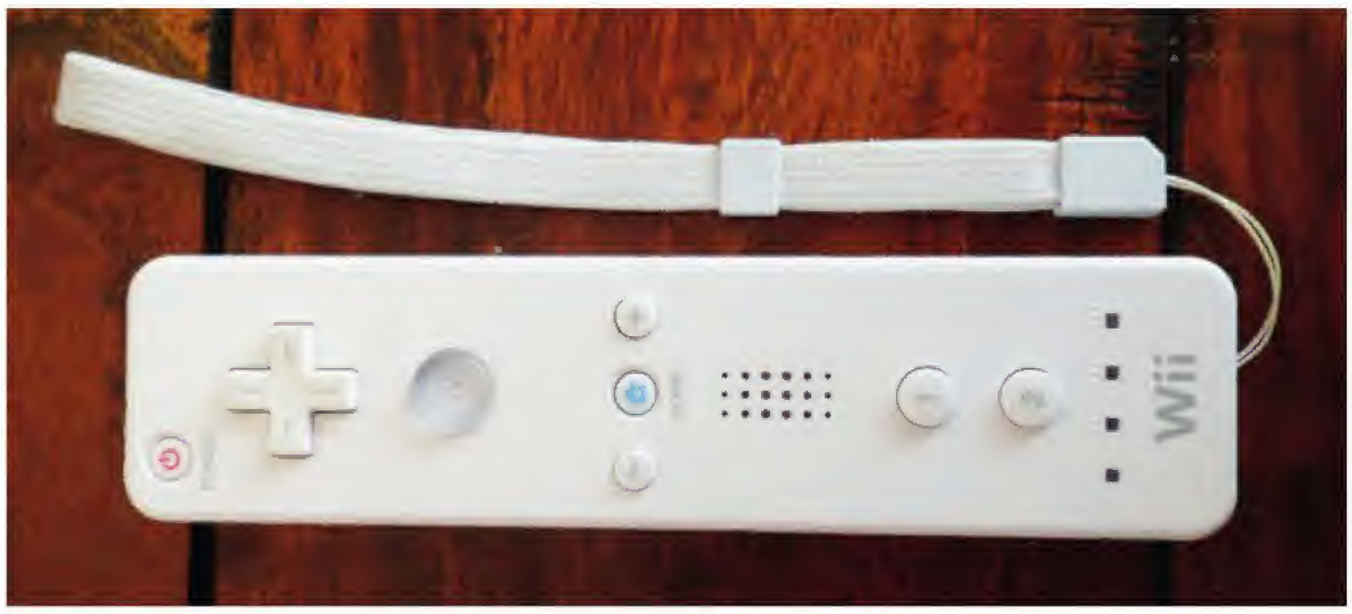
Our first line of code indented into the function runs the system function from the **os** library; this function enables us to run a shell command in Python, in this case the application **gpicview**. We then use string concatenation to join the filename **(selfie)** to the command, and to append the string with an ampersand, which is Linux shorthand for running a

command as a background process, releasing the terminal back to the user. The entire command is wrapped in a string **(str)** function, which formats the contents into a string.

```
os.system(str('gpicview '+(selfie)+(' &')))
```

We next instruct the function to wait for five seconds, giving our user time to view the picture; finally we run another shell command that will stop the picture viewer by killing its process:

```
sleep(5)
os.system('killall gpicview')
```



The Wiimote is a really cheap method of input that can be used with robotics as well as the Raspberry Pi Camera.

Bring it all together

With our functions complete, we now turn our attention to the main body of code. First we create a method to handle connecting the Wiimote to your Pi. We use a **try..except** construction to test that a connection is made. If there are errors, it will try three times to connect before exiting:

```
print('Press 1 + 2 on your Wii Remote now ...')
sleep(1)
wii = None
i = 1
while not wii:
  try:
    wii=cwiid.Wiimote()
  except RuntimeError:
    if (i>2):
      quit()
      break
    print('Error opening wiimote connection')
    print('Attempt '+str(i))
    print('Press 1 + 2 on your Wii Remote now ...')
    i = i + 1
```

Our next block of code is instructions to the user on how they use the selfie machine; this is simply a number of print functions with instructions on each line. You will notice that each line ends in **\n**; this is an instruction to Python to move to a new line:

```
print('Wii Remote connected...\n')
print('Press\n')
...
```

The final part of the main body is a **while True** loop that will constantly check the state of the Wiimote's buttons and save the value as a variable called **buttons**. We then create four conditional statements. Our first is a method to close the connection between the Wiimote and the Pi. By pressing the Minus and Plus buttons the program will exit:

```
if (buttons - cwiid.BTN_PLUS - cwiid.BTN_MINUS == 0):
  print('\nClosing connection ...')
  wii.rumble = 1
  sleep(1)
  wii.rumble = 0
  exit(wii)
```

If the user presses the Up button on the Wiimote, it will launch the **showpic()** function we created earlier:

```
elif (buttons & cwiid.BTN_UP):
  showpic()
```

If the user presses the A button, a photograph is taken. Remember the **pic** variable we used as an argument for the **takepic(pic)** function? Well here we create it by asking Python to save the current time and date in a YEAR, MONTH, DAY, HOUR, MINUTE, SECOND format. This is then printed in the shell, to show that it has worked. The **takepic(pic)** function is called and a selfie is taken!

```
elif (buttons & cwiid.BTN_A):
  pic = datetime.now().strftime('%Y-%m-%d-%H:%M:%S')+(".jpg")
  print(pic)
  sleep(3)
  takepic(pic)
  sleep(button_delay)
```

Our last condition handles the user pressing the B button, which creates the **vid** variable used as an argument in the **takevid(vid)** function we created earlier. The **vid** variable is almost exactly the same as **pic**: it records a timestamp for the video but replaces the **.jpg** with **.h264** instead:

```
elif (buttons & cwiid.BTN_B):
  vid = datetime.now().strftime('%Y-%m-%d-%H:%M:%S')+(".h264")
  takevid(vid)
  sleep(button_delay)
```

So that's it! Save your work and when ready click on Run > Run Module and have your Wiimote ready to test your selfie machine!

Code for this project

All of the code for this project is housed in a GitHub repository. GitHub is a great way to store your code so that it is readily available and backed up to the cloud. GitHub uses the *Git* version control framework to enable you to work on your code and then push it to the cloud; changes made on your machine can be pushed when ready, updating the code in the cloud. Others can fork your code and work on branches, for example creating new features. These are then submitted to you for approval, and when you're ready you can merge them with the main branch.

You can download the code for this project from **https://github.com/lesp/LV_Issue17_Education** if you are a GitHub user; if not you can download a Zip archive containing all of the files used from **https://github.com/lesp/LV_Issue17_Education/archive/master.zip**.

Les Pounder divides his time between tinkering with hardware and travelling the United Kingdom training teachers in the new IT curriculum.

LINUX VOICE TUTORIAL

SNAPPY UBUNTU CORE: NEXT-GEN PACKAGE MANAGEMENT

MIKE SAUNDERS

Discover a bunch of Ubuntu technologies that could define the Linux distributions of the future.

WHY DO THIS?

- Try a stripped-down Ubuntu variant
- Learn about transactional updates
- Understand how Snappy packages work

We all love to play around with new end-user features in Linux: new distro releases, updated desktop environments, and awesome graphical apps. But there's a huge amount going on under the hood in Linux right now, affecting the core of the operating system and the low-level plumbing that keeps it all ticking over. Some of these changes have been controversial and fiercely debated (such as *Systemd*), but one thing's for sure: Linux isn't hanging around. It's developing and modernising to be the best all-round platform for desktops, mobile devices and cloud deployments.

Now, a bunch of these new technologies have worked their way into Snappy Ubuntu Core, a new variant of the popular distro. We had a brief introduction to Snappy Ubuntu Core in issue 12's FAQ; here we'll look at it in more detail, explain why it's useful, and show you how its packaging system works. There's a lot of cutting-edge technology in this distro, but it's worth learning about as it could make its way into the mainstream distros we'll all be using next year. Even if you don't use Ubuntu yourself, given its prominent role in the Linux ecosystem it's important to keep track of developments.

1 UNDERSTANDING THE TECHNOLOGY

So, what is Snappy Ubuntu Core? First off, let's focus on the second two words: Ubuntu Core. This is a streamlined version of Ubuntu that weighs in at around 50MB and provides the bare essentials to get a Linux system up and running. You won't find any graphical desktops or web browsers here; it includes just the base system with the usual command line tools, libraries and hardware drivers. Additionally, it includes *apt-get* for retrieving extra software.

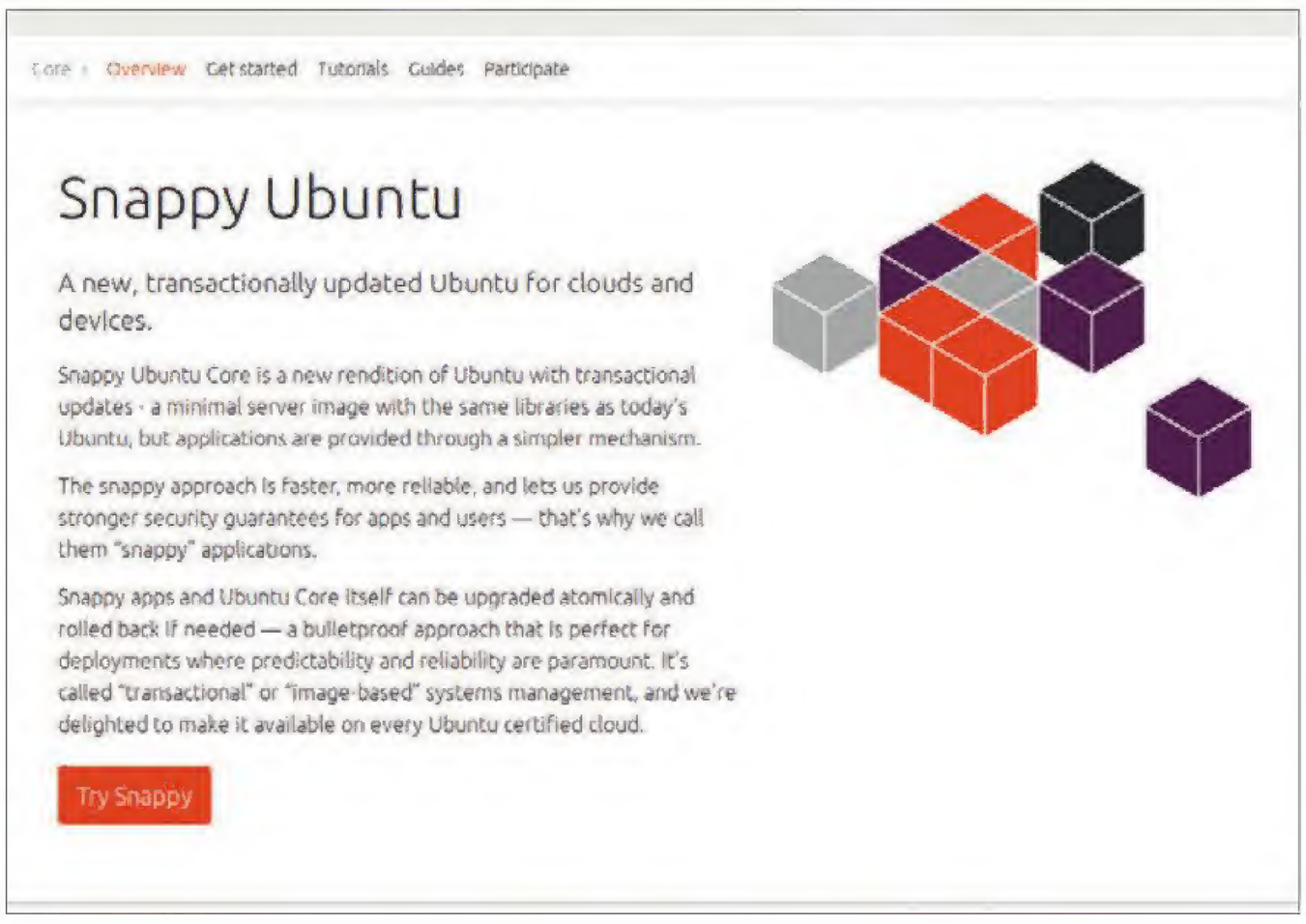
Ubuntu Core isn't designed for end users, but rather for distro and hardware developers. For instance, imagine you're designing a new single-board computer like the Raspberry Pi, and you want it to run Ubuntu. Instead of throwing the full desktop version of Ubuntu onto it, you'd opt for Ubuntu Core and customise it to your exact liking, choosing the packages and interface that fit the limitations of the device.

So far so good – but it's nothing special. There are a zillion trimmed-down distros out there doing the same thing. But this is where Snappy comes into play, which radically changes the way software is installed and how updates are applied. Snappy Ubuntu Core uses transactional updates, which means that they are either applied in full, or not at all. It also means that updates can be rolled back very quickly and easily.

The best way to explain this is like so: consider the current update mechanism in Ubuntu and other Debian-based distros. As root, you enter the following:

```
apt-get update && apt-get upgrade
```

This has worked pretty well over the years, but it has its limitations. What if you have a power cut or kernel panic during the installation of a certain package? The system files will be left in an undefined state. You may end up with executables from FooApp 1.2 installed, but libraries from FooApp 1.1. Configuration files may be out of sync. And if this is a system critical tool or library, what happens when you reboot? Your system may not come up properly. If you're willing to spend a lot of time, you could boot from a live machine, *chroot* into the broken installation and downgrade the offending package – if you can find it. It all becomes incredibly messy, very quickly.



Canonical is pushing Snappy Ubuntu as the next-gen distro for cloud deployments and mobile devices.

Transaction complete

Snappy Ubuntu Core aims to fix this by having transactional updates. Instead of having several

hundred packages for the base system (kernel, *Bash*, *glibc* etc.), this base system exists in a single package – so everything is updated at once. This base system is also provided in a read-only root partition, called partition A, and it also has an unused copy in partition B. When you update the base system, the working version in partition A is left alone; instead, the updates are applied to partition B.

On the next boot, the machine tries to boot from partition B, and if that works then partition B becomes the default, and the next round of updates, when available, will be applied to partition A. However, if the machine fails to boot from the updated partition B, it will revert to the known-as-working partition A. (Or if B boots but you have problems, you can switch back to A.) In this way, you always have a functioning operating system partition on your drive, and you can roll back to the previous state very quickly – without having to fiddle around with individual packages.

Now, a power outage during updates on a PC is very unlikely, but consider mobile devices. Canonical is pushing hard to get Ubuntu onto mobile phones, and you don't want those getting bricked because someone's battery ran out during a system update. We Linux geeks enjoy poking around in our OSes and fixing problems when they come up, but mobile phone users? They demand that everything just works, so a transactional update system with a simple rollback to the previous setup is essential – so that's why we have it.

2 TRYING IT OUT

Let's give this a go. In a terminal, grab the latest compressed filesystem image of Snappy Ubuntu Core from the net and extract it like so:

```
wget http://releases.ubuntu.com/15.04/ubuntu-15.04-snappy-amd64-generic.img.xz
unxz ubuntu-15.04-snappy-amd64-generic.img.xz
```

(This download is 120MB, and will extract to a **.img** file of 3.7GB in size.) Next, you need to boot this in a PC emulator – and the simplest option is to use *Qemu-KVM*. Find it in your distro's package repositories, install it and then run **kvm-ok** to check that it's working (you should see a message stating "KVM acceleration can be used"). Then boot it up in *Qemu-KVM* like so:

```
kvm -m 512 -redir :8022::22 ubuntu-15.04-snappy-amd64-generic.img
```

Here, the **-m 512** part says that we want our emulated PC to have 512MB of RAM, and the **-redir** bit redirects a network port on the emulated PC to ones on our host system. So once *Qemu-KVM* has finished booting Snappy Ubuntu Core, you can log into it from another terminal using:

```
ssh -p 8022 ubuntu@localhost
```

(We redirected port 22, the SSH port, from the emulated PC to port 8022 on the host machine, which is why you log in via localhost.) When prompted for the password, enter **ubuntu**. And that's it – you're running Snappy Ubuntu Core! You'll notice that there's not a lot going on here, as it's a very minimal installation. You can run commands as root by entering **sudo** before them, but note what happens if you try to run **apt-get** – you'll be told that this is a Snappy-only system. Yes, Snappy is the package manager of this distro, and is not only responsible for keeping the system in a bootable state as discussed earlier, but also for keeping programs well isolated from one another.

"Snappy is responsible for keeping programs well isolated from one another."

System upgrades

Enter the following command to view the filesystem layout of the virtual hard drive:

```
sudo cfdisk /dev/sda
```

You can see that the **/dev/sda3** and **/dev/sda4** partitions are both 1GB in size, and these are the read-only base system partitions (A and B) that we mentioned earlier. Press Q to quit **cfdisk**, and then enter:

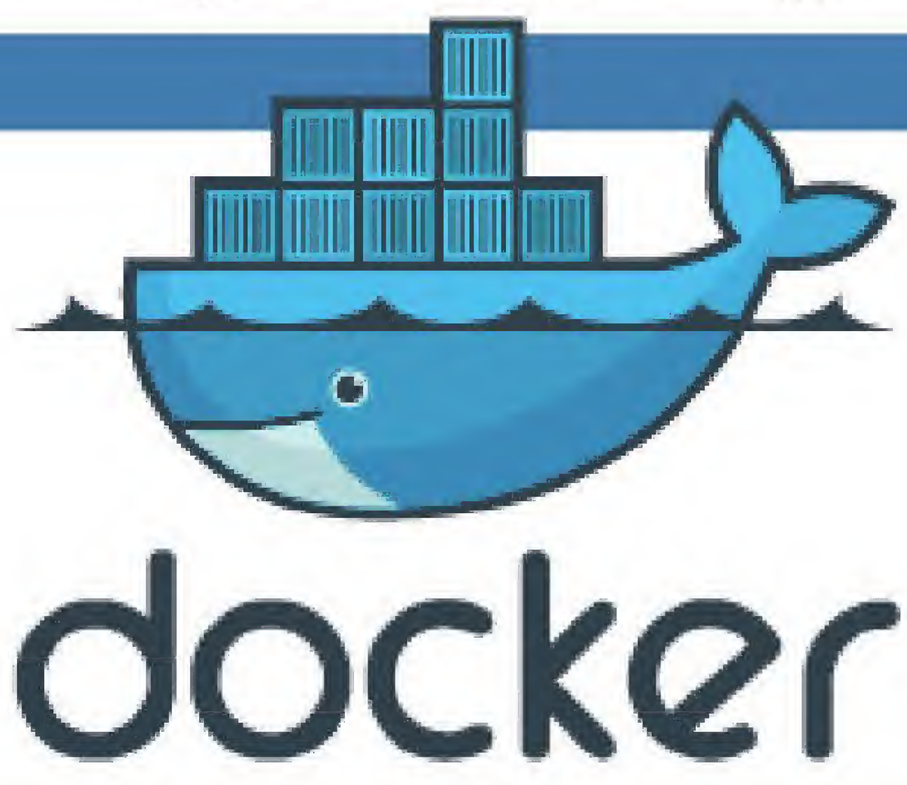
```
mount | grep /dev/sda3
```

Now you'll see that **/dev/sda3** is the root (**/**) partition,

Docking complete

Canonical is also pushing Snappy Ubuntu Core as "the perfect system for large-scale cloud container deployments". Ubuntu is already one of the most popular platforms for running *Docker* containers, and in Snappy it's available for installation with a single command (sudo snappy install docker). The idea with Snappy is that both the OS and the containerised software benefit from transactional updates and easy rollbacks. For more on *Docker*, check out our tutorial on p96 of issue 16.

Docker: just a fad, or the future of software distribution? Canonical is betting on the latter, and it's available in Snappy Ubuntu Core.



and the **ro** flag means that it's mounted read-only. So no programs can tamper with the base system – compare this with the case of normal Linux installations, where everything can be modified by processes running as root.

But what happens when you need to make changes to files in **/etc** and other directories? Well, repeat the above command but with **/dev/sda5** instead of **/dev/sda3**, and you'll see that it's mounted onto various places like **/etc**, **/var** and **/home**. It's also read-and-write, so this is where configuration files and user files live. Additionally, **/dev/sda5** is mounted onto **/apps**, which holds self-contained applications as we'll see in a moment.

So in summary: **/dev/sda3** contains the unchangeable base system for extra reliability and security, while **/dev/sda5** contains user-modifiable directories that are mapped on top – in other words, persistent data.

Going back to Snappy, enter this to see a list of packages installed on the machine:

```
sudo snappy list -v
```

You'll see that there aren't many in comparison with a typical Linux distribution. Pretty much everything is included in ubuntu-core. With Snappy, it's possible to have multiple versions of a package installed at the same time, and the active one is marked with an asterisk. To perform a system update, enter the following commands:

```
sudo snappy list -uv
sudo snappy update ubuntu-core
```

If an update to the **ubuntu-core** base system is available, it will be downloaded and installed into the alternative root partition – in this case, partition B (**/dev/sda4**). Upon next reboot, the system will attempt to boot from partition B. To revert to the previous version, use:

```
sudo snappy rollback ubuntu-core
```

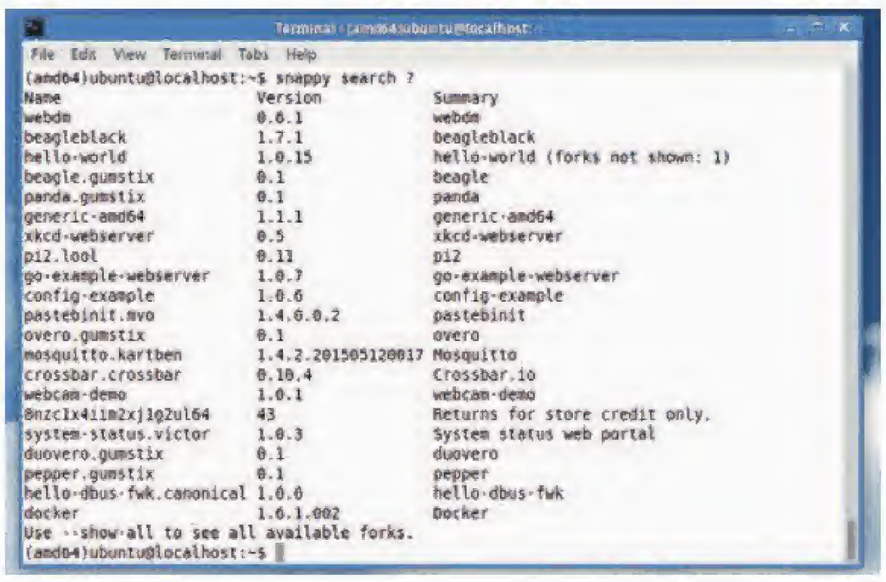
3 SNAPPY PACKAGES

So, what makes Snappy packages different to regular **.debs**? Most crucially, they are designed to be self-contained and not dependent on anything other than base system tools and libraries. They include everything they need, so they don't have lots of external library dependencies or are split up into many different packages for documentation, artwork and so forth. Ultimately, the goal here is to make software distribution quick and easy – especially for third-party app developers. You don't need to worry about what exact versions of everything are installed on a user's Ubuntu box; your program is pretty much guaranteed to work and doesn't care about the rest of the system.

Additionally, you can have multiple versions of the same program installed as mentioned earlier. To see how this all works, try installing an example Snappy package:

```
sudo snappy install hello-world
```

Now **cd** into the **/apps** directory, enter **ls**, and you'll see a directory called **hello-world.canonical**. Switch into that and enter **ls** again: along with a directory called 1.0.15, which contains the installed version of **hello-world**, you'll see a symbolic link called **current**. This link always points to the latest version, or if the user has rolled back a package after problems with an



Use snappy search to find packages – but note that most of them are demos or hardware drivers right now.

Snappy vs RPMs/Debs: the trade-offs

With traditional Linux package management systems, dependencies play a major role. A single program can depend on hundreds of other packages – libraries, toolkits, artwork, documentation and so forth. Snappy packages, in contrast, have much more in common with software distribution on Windows or Mac OS X: everything that the program depends on should be distributed with the program itself.

There are upsides and downsides to this. On the upside, it makes it very easy for third-party developers to distribute software: users can install a Snappy package and it will almost certainly work. The package doesn't care which libraries are installed on the system, or where they are, or which versions they are. An update to Libfoobar which subtly changes the behaviour of one of its routines can't break other software in mysterious ways, for instance.

On the downside, this introduces problems with security. Take a library that's used by many different programs, such as *Zlib* (for compression). With traditional Linux packaging systems, there's one copy of *Zlib* on the system, and if a security hole is discovered in it, only that package needs to be updated. All programs that depend on the library will automatically be fixed.

If every program starts bundling its own copy of *Zlib*, however, it gets complicated: if a security hole is discovered, every program needs to be updated independently. Some developers will respond quickly to security holes – others may take longer, or not bother to fix. As the user or sysadmin, it's difficult to tell which is which, and whether your system is safe. Additionally, with every Snappy package bundling all of the libraries it needs, this takes up more disk space.

update, it will point to a previous version. This is how multiple versions can live in a Snappy Ubuntu Core installation side by side.

Enter **cd 1.0.15/bin** and run **./echo** to execute the main program in **hello-world** – and as you'd expect, it prints "Hello World" to the screen. Enter **ls** and you'll see other executables in the directory, such as **env** and **showdev**. You can run these executables from any location in the filesystem by entering the name of the package, a full stop, and the executable, eg:

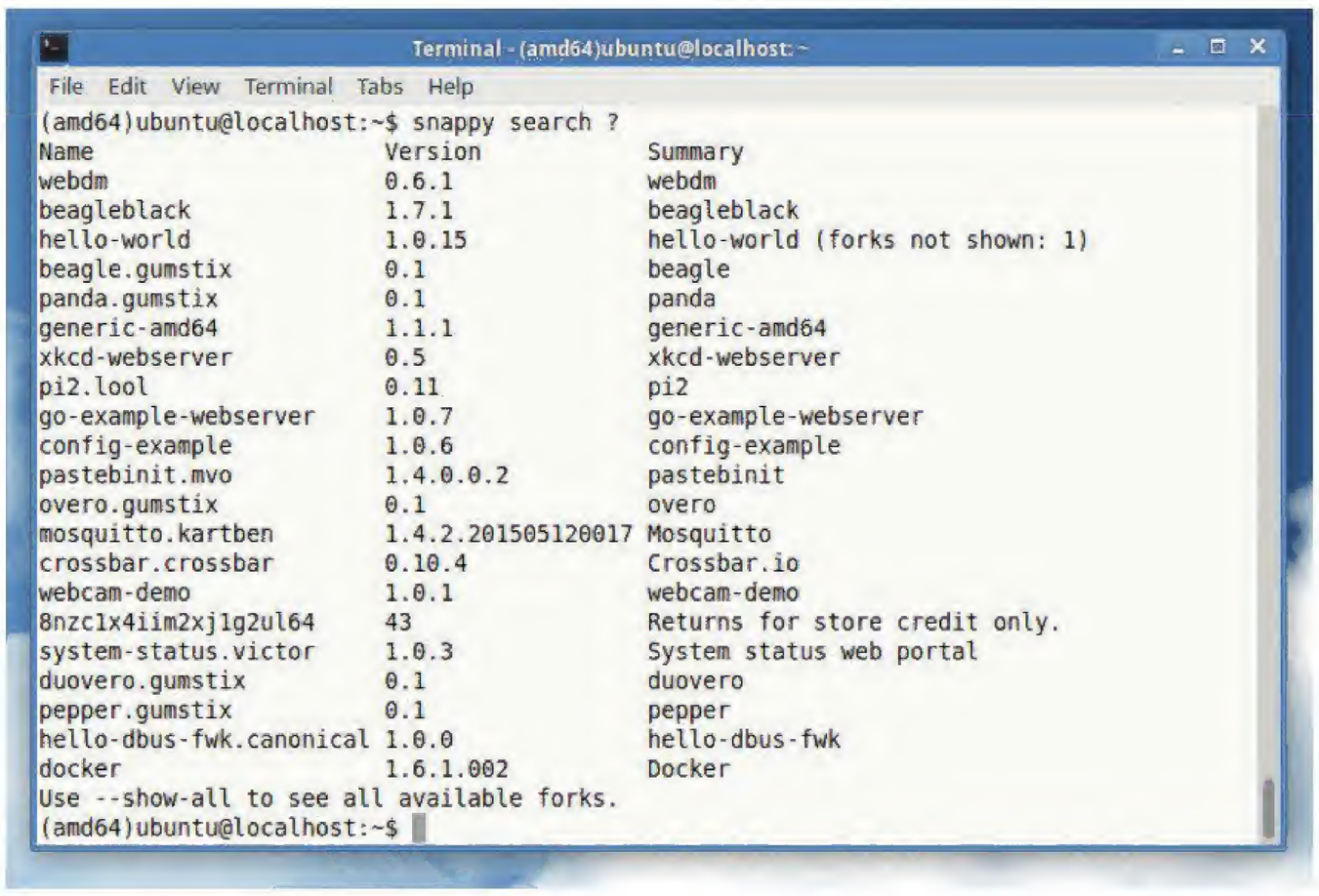
```
hello-world.echo
```

But how does the system know where to find these executables? If you look at your **$PATH** (eg **echo $PATH**) you'll see that there's no entry for **/apps/hello-world.canonical/1.0.15/bin**. If the **$PATH** needed to be updated for every app you install, it would become unwieldy. So instead, scripts are added to **/apps/bin** whenever an app is installed, which call the relevant programs. Have a look at **/apps/bin/hello-world.echo**, for instance, and you'll see a chunk of boilerplate Snappy code to set up the execution environment, while the last two lines execute **/apps/hello-world.canonical/1.0.15/bin/echo**.

Snibeti Snab

Switch back into the **/apps/hello-world.canonical/1.0.15** directory and enter **ls** again, and alongside the **bin** directory you'll also see one called **meta**. As you'd expect, this contains metadata for the package – in other words, not the software itself, but information describing how it works. Switch into the **meta** directory and have a look around; you'll see that there's a file called **package.yaml**, which contains the most important information about the package (name, version, vendor, icon and so forth). This **package.yaml** is one of two mandatory files in a Snappy package, the other being **readme.md**, a description of the software in Markdown format.

For tighter security, Snappy packages are also restricted in their capabilities using *AppArmor*, a Mandatory Access Control (MAC) system that, among other things, stops executables from being able to access certain files. Each executable in the **hello-world** package has an associated **.apparmor** file in the



Snappy package metadata is provided in YAML (Yet Another Markup Language – **www.yaml.org**) format.

meta directory, so if you look inside **echo.apparmor** for instance, you'll see that it's assigned the "default" *AppArmor* template. In other words, it runs with default permissions. With *AppArmor*, it's possible to restrict programs from accessing certain network resources and filesystem locations – useful if you're installing software from an untrusted third-party source.

As with operating system updates, Snappy package updates are transactional so they can't leave you with a broken setup. Because newer versions are placed in different directories inside **/apps/appname**, you can always revert to an older version if you come across any bugs. If you've ever tried to have multiple versions of the same program installed on your Linux box, you'll know just how difficult and messy that can be – so this is a welcome development. Ultimately, users will have the freedom to try newer releases without having to overwrite their old software and potentially messing up the system.

"Canonical has hinted that Snappy packages could become part of the mainstream distro."

So that's an overview of Snappy Ubuntu Core, exploring how its update and packaging systems work. Much of the technology is still undergoing heavy development, but Canonical has hinted that Snappy packages could become part of the mainstream desktop distribution in the future, and if it works well, we could potentially see it adopted by other distributions as well. For more on Snappy, see Canonical's documentation at **https://developer.ubuntu.com/en/snappy**, and if you'd like us to cover any aspect of it in more details (such as creating Snappy packages by hand), drop us a line!

Much of the technology behind Snappy comes from Canonical's work on its Ubuntu Phone.

Mike Saunders has seen the future, although he has no idea what next week's lottery numbers will be. Sorry, everyone.

LINUX VOICE TUTORIAL

RUN DOS AND WINDOWS GAMES ON LINUX

GRAHAM MORRISON

Broaden your games collection with a few classics from those other forgotten operating systems.

WHY DO THIS?

- Play some awesome games for free or cheap
- Brush up on essential cross-platform skills
- Work with Windows binaries on Linux

Thanks to Valve and its decision to switch from Windows to Linux, our favourite operating system is in the process of becoming a major gaming platform. Valve's games portal, Steam, currently lists over 1,100 Linux titles, including blockbuster releases like *Team Fortress 2*, *Portal 2*, *Borderlands 2*, *The Witcher 2*, *Bioshock Infinite* and many more. This would have been unimaginable a couple of years ago, and it's likely to get even better as Valve works towards releasing its own Linux-based platform to compete with games consoles. It's the reason why many of us are looking at upgrading our machines and moving them closer to the television.

Alongside the shiny new native version of games that run on Linux is a vast library of classics yet to be played, and almost any modern machine can play them – even the humble Raspberry Pi or your Android phone. The only slight hitch is that while these games can run on Linux, it's not how they were intended to run. These are games that were typically made for older versions of Microsoft Windows, and before that, Microsoft DOS. But we have the power to get these games running on your Linux desktop, and that's without running a virtual machine or needing a Windows licence. So raid your shelves, grab those games you've not played for a decade and spend a wet afternoon enjoying some classics.

Even though there's now a native version of Steam, and natives versions of many of its games, you get access to many more when running the Windows version.

1 DOSBOX

Before Microsoft Windows (and Linux) there was DOS, the classic command prompt interface that turned generic PC hardware into something useful. In the 1990s, DOS – the Disk Operating System – became a game programmer's playground because it gave direct access to hardware with rapidly accelerating performance and an equivalent drop in price, especially when compared with the Apple or Commodore computers of the same time. All this new processing power led to the development of new types of games, such as *Doom*, as well as what became 3D acceleration devices for gaming, and there are all kinds of classics from this period. Remarkably, you can play thousands of them directly from your browser while they're being held in cold storage at **https://archive.org/details/softwarelibrary_msdos_games**.

Take a trip down memory lane...

The software that **archive.org** uses to drive its in-browser gaming engine is called *DOSBox*, which is itself a brilliant GPL-licensed problem solving tool capable of running far more than just games. *DOSBox* is essentially an emulator for those earlier machines, in much the same way *CCS64* emulates all the hardware intricacies of a Commodore 64. But while *DOSBox* can and does emulate the hardware environment of those earlier machines, it can also pass CPU instructions on to your native processor for massive speed and efficiency gains. This is because the x86 CPUs used by most computers are still derived from the CPUs in those early PCs and share many of the same features and instructions. Unfortunately, you don't get the same boost from a different architecture such as the Raspberry Pi or ARM running Android, but you at least get the same compatibility.

DOSBox is a simple point-and-click install from your favourite package manager. But like those early DOS environments, *DOSBox* can be a little cryptic to get running usefully. The first step is running the

Daggerfall **is one of the best RPGs made, and it's a free download from the original publisher.**

executable. This will transport you from the modern world of social networking, pervasive data networks and clouds, and drop you into the world of Sound Blaster, **HIMEM.SYS** and IRQ assignment. The first line is already typed for you – **SET BLASTER=A220 I7 D1 H5 T6**, configuring audio playback variables for maximum compatibility, and you'll notice the command prompt flashing **Z:\>**.

DOS is similar to the *Bash* command line, with some important differences. The **ls** command is replaced by **dir**, for example, while **cd** will still change directory. The internal mounted virtual drive is known as Z, containing the simple tools required to boot most DOS applications – you can see a list by typing **dir**. To do something meaningful, we'll need to create a portal between *DOSBox* and our files back in the real world. You can do this by typing the following:

```
mount c /home/graham/games  -freesize 1000
```

We created a folder in our home directory called **games**, so you'll need to modify the above command for your own installation. The **freesize** argument is there because the storage capacity of our modern drives is far in excess of what DOS is expecting, so we're pretending our folder is far more modestly capable. In this case, we're mounting the folder and providing just 1000MB of storage, which is still more than enough for a CD installation. In *DOSBox*, you can change to this new drive by typing **C:**, and you can check its contents by typing **dir** – experience the nostalgia of filenames limited to eight characters.

If you have the contents of your games media handy, such as the files off a CD-ROM or a floppy drive, you can move them to your mounted folder and access them just as you would the original media. However, we're going to use an ISO image of one of our favourite games – *Daggerfall*. This is a brilliant 3D RPG and a forerunner to both *Oblivion* and *Skyrim*. *Daggerfall* is available as a free download from the publisher (see **www.elderscrolls.com/daggerfall**). This download is a 'rar' archive that can be extracted into your games folder and acted upon just as if you'd mounted an ISO or optical drive. But as most games you'll own can be turned into an ISO, and because we own *Daggerfall*, we'll include this step too. Whenever you make changes outside of *DOSBox* to a mounted folder, you need to press Ctrl and F4 to refresh *DOSBox*, otherwise they won't appear. You can then mount the ISO image from within *DOSBox*, and you do this with the following command:

```
IMGMOUNT D DAGGER~1.ISO -t iso
```

In the above line, the real filename of the disc image, **daggerfall.iso**, has been truncated with the **~1** symbols to accommodate the length restriction on filenames in DOS. To get around the awkwardness of guessing and typing these names, just press Tab to get name completion after entering the first few letters. With the above command executed, you'll find the contents of the ISO disc image hanging on the end of the **D** drive designation, which we can switch to by typing **d:**. If you've extracted a rar archive, just **cd** into its **DFCD** folder. The contents of both the ISO and the folder will be the same.

Installation

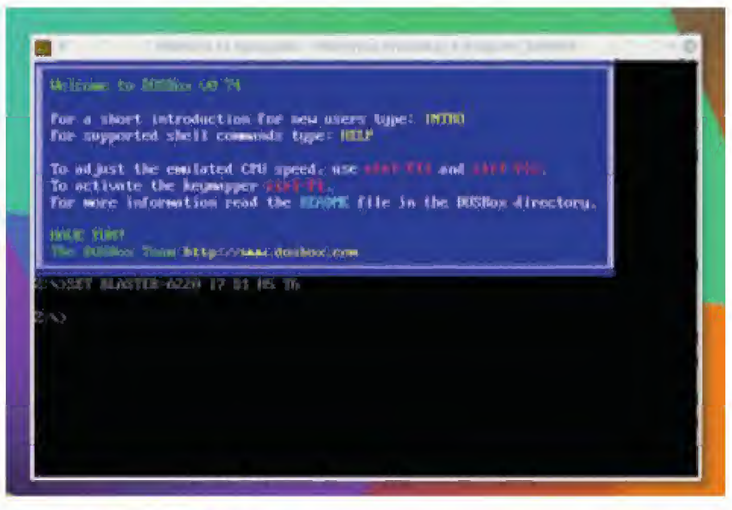
By 1995, the price of a Seagate hard drive the size we created earlier (1000MB) was approximately $850 – expensive but worth the outlay for most computer users. Storage was becoming affordable, and for that reason, most mid- to late 90s DOS games will need installing. In an age before standard toolkits, most installers were different. You will need to briefly check any **README.TXT** files, documentation or even the manual, if you've still got it. Most, including *Daggerfall*, use their own installer, an **install.exe** executable that's run by typing **install** from the directory. Installers will enable you to select an amount to install and a destination (you're usually best off sticking with the defaults). After those files have been copied, you'll also need to negotiate soundcard configuration. *Daggerfall* does this too, and thanks to the **SET** command that runs when you launch *DOSBox*, selecting 'Auto Detect' should work. If

"DOS is similar to the Bash command line in Linux, with some important differences."

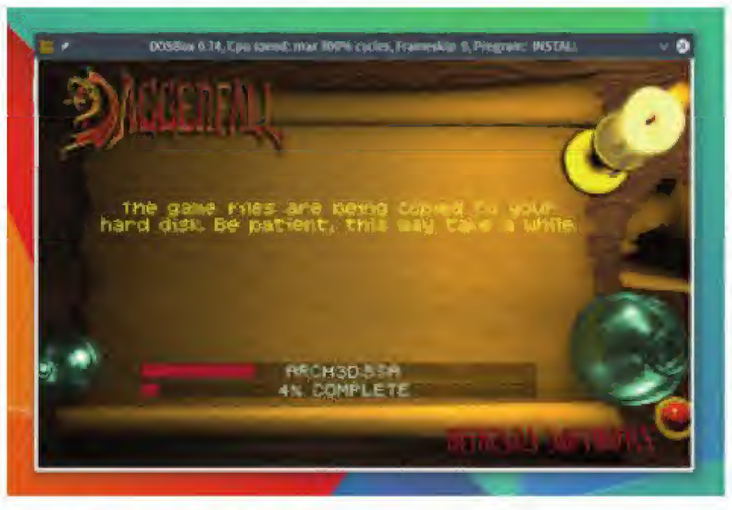
LV PRO TIP

List all the commands supported by *DOSBox* by typing **help /all**.

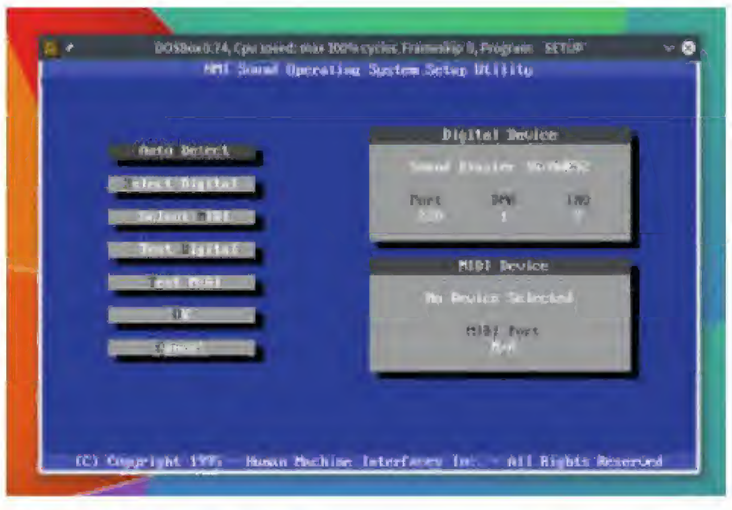
Step by step: Installing DOS games



1 Start DOSBox
When you run *DOSBox*, it will prepare the environment for running your games. You'll need to mount and navigate to the installer's mounted folder.



2 Run install.exe
The installer is usually called **install.exe**, and they're usually unique to each game. Here we're performing a full install of *Daggerfall*.



3 Configuration
All DOS games will require you to enter the details of your sound device, but auto-detect should almost always work out fine.

not, the default settings are Sound Blaster 16/AWE32, Port 220, DMA 1 and IRQ 7. After exiting any installer, you can usually change these settings by running **setup** or by editing a configuration file (usually ending with **.cfg**). One final step required by *Daggerfall* and many other games too, is to run an update. We downloaded ours from the publisher, and you can usually find updates for popular titles. The update was a **dag123** executable, which we ran manually from the installation folder, after which we could type **dagger** to finally run the game. The name of the executable should always be obvious and is usually output by the installer. Just type this to run the game. Many games also expect to have the installation medium installed in the same location, such as the ISO mounted, as this was their form of copy protection, and you'll need to make sure this is in the same location each time you run the game.

LV PRO TIP

If you're running *DOSBox* in a window and you want to get your mouse back to the desktop, press Ctrl and F10.

2 RUNNING WINDOWS GAMES

While DOS games are lots of fun, many people have a larger collection that require Microsoft Windows, especially if you migrated to Linux from Redmond's own OS. Many of the second-hand games you're likely to find are going to be from the last decade, rather than the last two decades, which also makes Windows rather unavoidable.

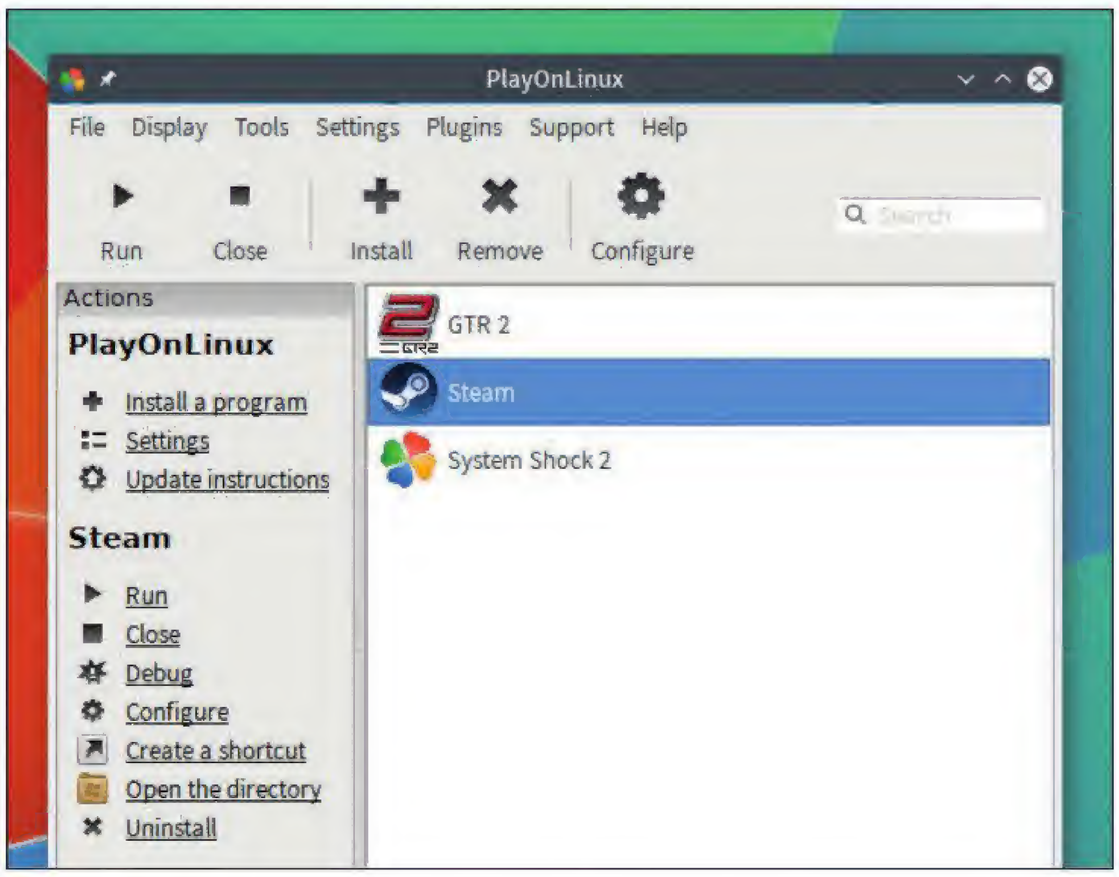
There are several solutions, with perhaps the easiest being to install a legal copy of Windows into a virtual machine. With this, you will get a perfect recreation of the operating system, but you won't get any meaningful hardware acceleration to help with game performance (although you will get some) and you'll need a licence to use it. Virtual machines like this are very important if you need 100% compatibility for some Windows software, but there's a better open source option that works for many games and applications, and that's called *Wine*.

Wine calls itself a 'compatibility layer' rather than an emulator (*Wine* is an acronym for Wine Is Not an Emulator, to make the point). Instead of emulating hardware, such as recreating the sound chip of a Commodore 64, *Wine* recreates what Windows applications expect from the operating system, usually by replacing Windows functionality with Linux functionality. This functionality is augmented with Windows tools that are downloaded and installed alongside, just as you would on Windows itself, such as DirectX for accelerated graphics, codecs for video playback and fonts so that text looks the same. It can get extremely complicated, and the configuration and maintenance of a working *Wine* environment can take some time and effort, especially when changing settings can affect compatibility.

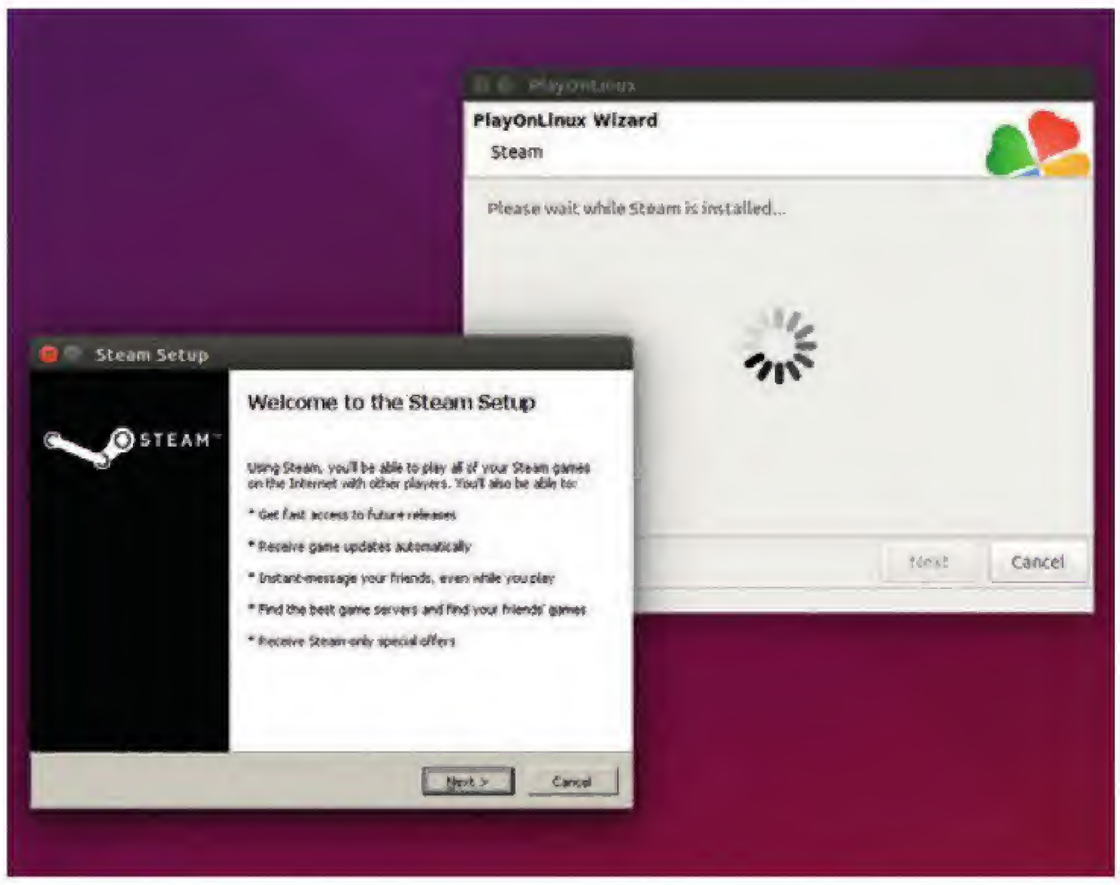
PlayOnLinux

Fortunately, there's an easy solution: *PlayOnLinux* is a wrapper around *Wine* installations and their configuration for specific applications and games. It makes installing something like a Windows game much easier. Most distributions include the *PlayOnLinux* package, and the package itself will handle *Wine* downloads, so you won't find this as a dependency. This is because specific versions of *Wine* are tested with specific games and applications, and the developers will only support versions they know work well together. For that reason, *PlayOnLinux* will juggle several versions of *Wine* installed at the same time, and it also means that installing the average title will take a considerable amount of network bandwidth as it downloads the prerequisite fonts, libraries and DirectX version for each successive version of *Wine*.

When first launched, *PlayOnLinux* will download the latest list of supported titles. Click on 'Install' and



Games are installed into their own virtual drives to avoid cross-contamination of configurations and *Wine* versions.



PlayOnLinux will cleverly download any missing parts of Windows as required by whatever game or application you're trying to install.

switch to the Games list to see what's available. There are hundreds of compatible titles, including digital downloads and games from CD and DVD. You'll even find some **GOG.com** titles listed, adding Linux support to your downloads from the main site, and the Windows Steam client can also be installed, adding many more titles that are yet to make it to Linux. We installed both the Steam client and our favourite old classic, *System Shock 2*, which we own on CD. In both cases, you're guided through the installation via on-screen instructions.

Installing from CD

For *System Shock 2*, we needed to insert the CD before we started. This is because the first question you're asked by *PlayOnLinux* is where the optical drive is mounted, and it will list the mounted volumes it detects. Hopefully, one of these will be your drive. If not, use the 'Other' list item to point the requester at the location of your mounted drive or ISO image. The Windows installer will then be launched from the drive and you'll need to run through the installation options for your games, including the entry of serial numbers if this is used to protect against copying.

If you get the option, we'd recommend choosing a 'Full Install' so that as many files as possible are copied off the installation medium. We needed to use Tab on the keyboard to select between some options in the Windows installer, and *PlayOnLinux* should also notice when other packages are installed. Most will need fonts and DirectX, and *System Shock 2* needed an Intel media codec for video playback. You may also need to Alt Tab to hidden windows if the installer stops. Finally, you'll be asked for your graphics card's configuration, which is usually just the amount of RAM on board. The Steam client required a few more steps because the client itself needed to be launched several times to enable a few updates to be automatically installed, but *PlayOnLinux* successfully navigated the complicated third-party packages that it needed and installed them at the same time.

After games and applications are installed, they'll appear as desktop icons (if you let them) and within the main *PlayOnLinux* application window. Launching them is now just a click away, and they should perform almost as well as their natively installed counterparts. From the Steam client, you can install other titles but you'll still find compatibility problems with the latest releases.

"Games and applications will appear as desktop icons and within the PlayOnLinux menus."

If you want to make changes to the Windows installation of a game, right click on its entry within *PlayOnLinux* and you can choose to open the directory where the application/game is installed. This is the place in your filesystem where the Windows files are installed and is usually isolated from other games to maintain compatibility. Opening the folder is useful if you need to manually add updates to an installation, such as replacing an executable for a games update. You can also configure each *Wine* installation from the same menu. *Wine*'s configuration panel enables you to change the location of the optical drive and the graphics resolution of the display, as well as the version of Windows that's being mimicked. This is usually Windows 98 for maximum compatibility, but you can choose anything between the ancient Windows 2.0 and Windows 8.

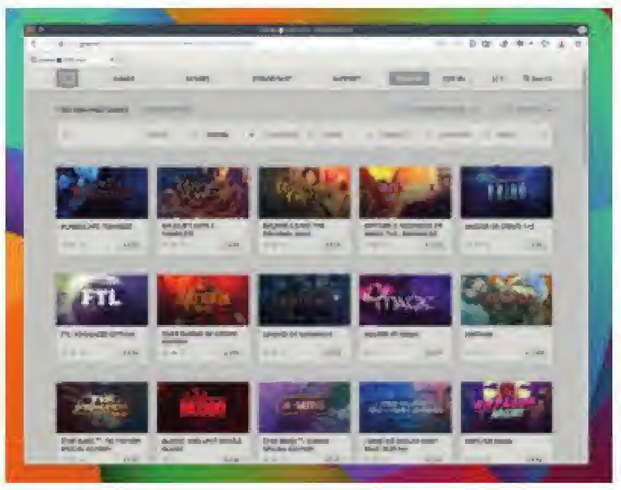
Graham Morrison is the editor of Linux Voice, a lapsed KDE contributor and a collector of old synthesizers.

Games sources

Buying pre-owned games is big business. Whether it's from your local store or eBay, there's a huge selection of classics most of us have missed. There are also many sites of dubious legality offering access to what they call 'abandonware', as well the stack of games hosted at the Internet Archive.

GOG.com (formally Good Old Games) has turned playing old classics into a business model, offering older games at a lower price, and packages together with tools like *DOSBox* to enable configuration-free playback. What's more important is that it works with the original publishers to remove any DRM, which is often a stumbling block when playing old games and much more likely to help those games work with Linux. Which is perhaps why beta Linux support is now available at **GOG.com**.

If you're serious about playing old games, you might also want to look at *CrossOver*. This is a commercial version of *Wine* that uses a similar profiles system to *PlayOnLinux* to create point-and-click installers for many games and applications. You could even try the free demo if you wanted to check compatibility first. The great thing about *CrossOver* is that it's also a major contributor to the *Wine* project itself, as all developments made to the commercial version are merged into the open source version, helping the development of both products.



GOG.com is a games distribution service primarily for older games, and now offers Linux support.

Getting CDs and DVDs onto your computer

Many of those old CDs and DVDs holding games are likely to be scratched and separated from their wallets, which means their time as effective data sources is limited. Now that storage is so cheap, it makes good sense to move them onto your hard drive. The simplest method is to copy all the files and folder structure over from the mounted optical drive into a new folder on your machine. You can do this from the command line or from a file manager, and as long as you point *Wine* or *DOSBox* at the location of this root folder to use as the optical drive, it will work fine.

A neater solution is to create an ISO image of the disc. This is a single file that contains both the files and the filesystem of the disc, which can help with some game compatibility when the game is checking for whether a disc is inserted. You can create an ISO using a GUI tool like *Brasero* by using the 'Disc Copy' option, or from the command line with the **dd** command. If you're going to store an ISO image, use **7zip** to compress them as it can also decompress an ISO's contents. You can access an ISO directly with *DOSBox*, or mount it onto your filesystem with the following command (**/mnt/iso** will need to exist first):

```
sudo mount -t iso9660 -o loop cd.iso /mnt/iso/
```

LV PRO TIP

With *Wine* installed, you can run simple Windows binaries by typing **wine** followed by the name of the **.exe** file.

LINUX VOICE TUTORIAL

IMAGING IN THE RAW WITH NEAR INFRARED

ANDREW CONWAY

Still have your camera hooked up to your Raspberry Pi? Good – put it to good use and learn some science at the same time.

WHY DO THIS?

- Make accurate measurements from images – do amateur science
- Understand your camera, get beautiful images
- See in the infrared

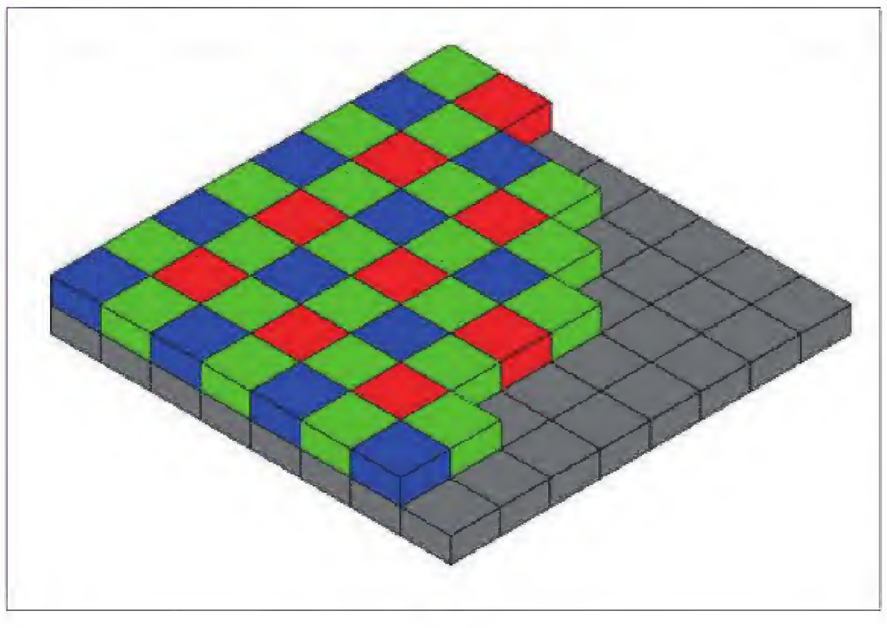
In the heart of the Sun, nuclear fusion reactions produce energy. This leaks out to the surface of the Sun where it is liberated in a huge number of photons – particles of light. A photon can traverse 150 million km unimpeded, make it through the Earth's atmosphere, through a chink in your curtains and strike your face in the morning. Then when you open your eyes, an enormous number of photons will enter them every second enabling your brain to construct an image of the world you see.

Our eyes are remarkable. What's even more remarkable is that for under £20/$20/€20 you can buy a device that does a similar job that plugs into a Raspberry Pi. In some ways it's even more sensitive than the human eye in that it allows precise measurement and, if you buy the Pi NoIR version, it works in the infrared.

We're going to take a look at raw data produced by a camera and understand how it can be processed for a number of purposes, both aesthetic and scientific. Don't worry if you don't have a Raspberry Pi, because much of what we'll explore can be applied to raw data files from any camera.

Bayer pixels

At the back of a camera is a grid of light sensors that you can think of as hardware pixels. Each one records the amount of light that enters it and this is read by the electronics as an integer, which will end up in the raw image file. However! These hardware pixels are not sensitive to the colour of light. So to give us colour images, each one of these pixels has a tiny filter placed on top of it. In a typical modern camera half the pixels have green filters, a quarter have red filters and a quarter have blue filters, arranged as shown in the image below.



Bayer pixels are arranged on the camera sensor in groups of four with two green, one red and one blue pixel in each 2 by 2 group. Green is over-represented to emulate the colour response of the human eye, but it's also due to geometry and efficiency of manufacture.

This arrangement is called the Bayer pattern, after the person who first proposed it. The total number of these pixels is usually the same as the advertised image size of your camera; so if your camera boasts 8 megapixels, which is about 8 million, then that is the number of Bayer pixels. In your final image, say a JPEG, each pixel will be represented by a colour made up from mixing red, green and blue values, eg an RGB of (255,0,0) is red, and (255,255,0) is yellow. Since a Bayer pixel only records the intensity for one colour, the colours need to be estimated from surrounding pixels. The process of doing this is known as demosaicing or interpolation.

In images with gentle colour gradients, such as a view across a grass field, the artefacts from demosaicing will not be too noticeable. They may however become apparent if there is sudden change in colour and intensity, say around the edges of a window. More powerful CPUs and GPUs and advanced interpolation reduce such problems, but the best solution is to increase the pixel resolution.

Usually we want our photographs to look natural, that is, to resemble what we'd see with our own eyes. To achieve this we must balance the colours after demosaicing the Bayer pixels – a process known as white balancing. Using the red, green and blue values as reported by the sensor will almost certainly result in odd-looking colours. For example, a white object might appear slightly bluish. White balance varies from camera to camera, and depends on exposure and lighting conditions. Although acceptable defaults and algorithms for balancing are shipped with most camera firmware, professional photographers will often want to take control of the colour balance for the best results.

Having established that demosaicing and white balancing are important for most photographs, let's now turn to an example where they would destroy information, and working with raw data is a must.

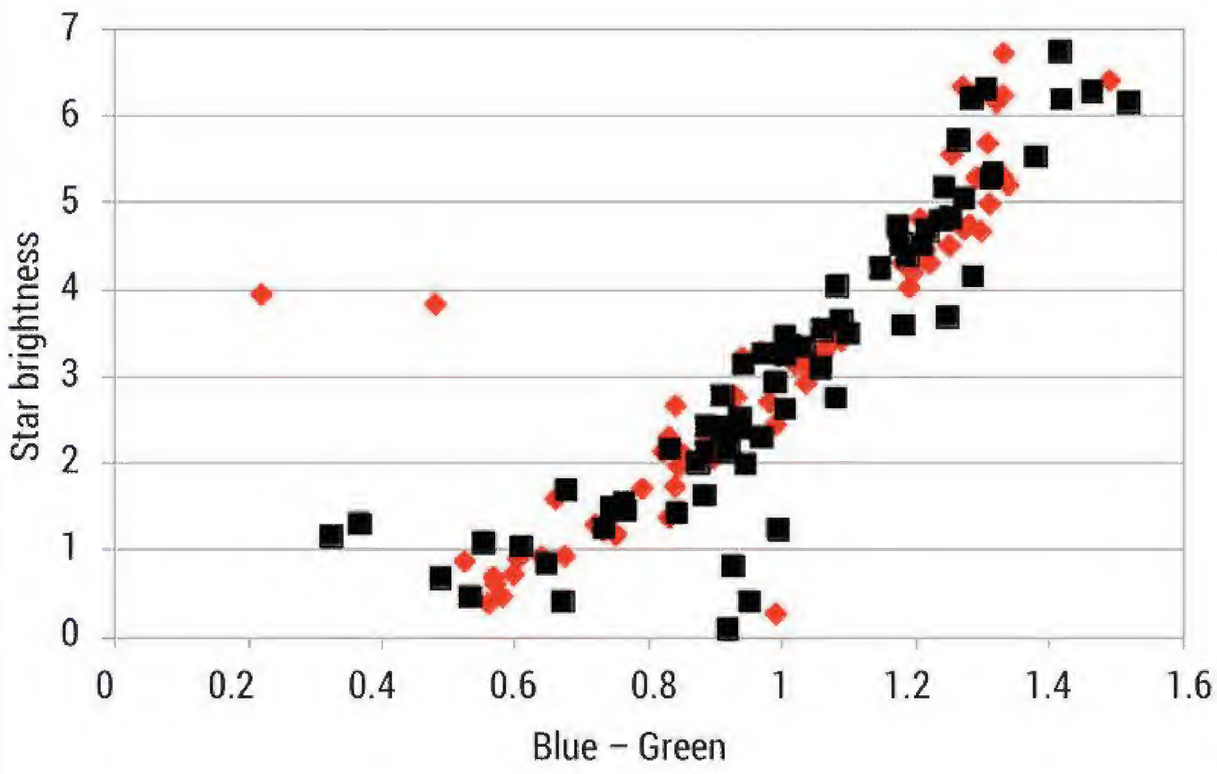
Amateur science

Working with raw data enables you to make quantitative measurements from your images. This has applications across all branches of science, but we'll take a look at an astronomical one first.

It's possible to use a normal digital camera, or the Raspberry Pi camera, to do useful astronomical work. The simplest question you can ask about a star is: how bright is it? For this you must work with the raw

Star cluster colours

Here's the star cluster M44, which you can see with the naked eye in the constellation of Cancer. The graph shows data obtained by amateur astronomer George Johnston on how star colours correlate with brightness – known as the Hertzsprung-Russell diagram. The black squares are the values George obtained using a Canon EOS 550D camera, and show the same correlation as the red diamonds from data published by professional astronomers (the trend is for brighter stars to be bluer). The scales are logarithmic but run the opposite way to the magnitude system used in astronomy.



(image credit: Miguel Garcia)

data, because any attempt at demosaicing the Bayer pixels will distort the results.

Without even using a telescope you can measure the changing brightness of a variable star such as Algol. Put a normal digital camera on a tripod, take a few seconds of exposure and then measure the brightness of a star by adding up the values of all the pixels its image covers. Do this over a number of nights and you'll be able to plot a graph of Algol's variability over time.

It's even feasible to discover a planet orbiting a star by looking carefully at the star's brightness. If the star has a planet and its orbit is aligned so that it passes between the star and the Earth, then a dip in starlight might be detectable. Of course, being feasible doesn't mean it's easy: a decent telescope would be needed, as well as a lot of patient searching and honing of technique.

Stars have different colours, with blue stars being hotter than red stars. In the early days of astronomy, the colour of a star was quantified by putting a blue filter at the end of a telescope and taking a brightness measurement and then repeating the same measurement but with a different filter. The difference in brightnesses was called the colour index. The Bayer pixels in a modern digital camera can be used in much the same way – see the boxout above for an example of this.

The Raspberry Pi camera

Let's play with data ourselves using a Raspberry Pi camera. If you want to work with raw data from some other camera then you can skip to the next section. Alternatively, if you'd like to use the exact same data as I'm using, then you can get my raw images of trees along with the Python source code from **https://github.com/mcnalu/linuxvoice-imaging**.

There are two ways we can work with the Raspberry Pi camera: either from the Linux command line, or using Python. We're going to use the first, but if you want to use Python then have a look at the **PiCamera** module, and in particular the instructions on Raw Bayer capture that can be found here: **http://picamera.readthedocs.org/en/latest/recipes2.html#raw-bayer-data-captures**.

First, make sure your Raspberry Pi is set up with its camera and ready to use. Next, point the camera at something interesting and colourful, open up a terminal and enter the following:

"It's even feasible to discover a planet orbiting a star by looking at the star's brightness."

```
raspistill --raw -o image.jpg
```

This will take a picture and save it to the file **image.jpg**. The **--raw** option means that the raw information from the camera sensor will be embedded in the file.

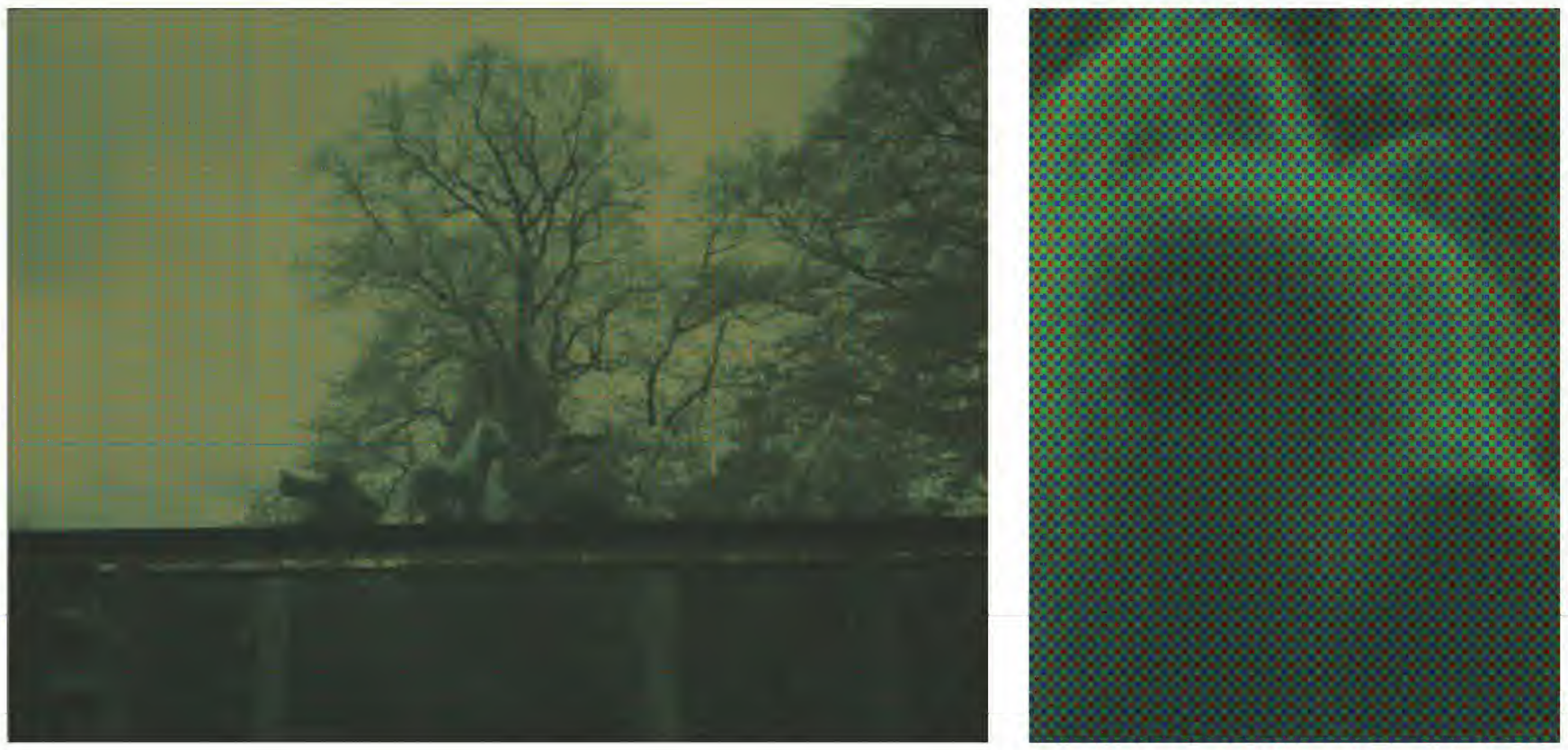
Now we have the image data, you could continue to work on your Raspberry Pi, but it'll probably be faster to copy **image.jpg** to a desktop or laptop computer running Linux to perform the raw data extraction and processing. Next, we'll need to extract the raw data from the **.jpg** file. To do this we'll need to make use of a nifty utility called **raspiraw**. It's not available in distro repositories, so we'll need to pull its source code (just one C file!) from GitHub and build it:

```
git clone https://github.com/illes/raspiraw.github
cd raspiraw
make
```

Copy **image.jpg** to the **raspiraw** directory and perform this command:

Everything's gone green

This image shows the raw data captured from the camera sensor. Each pixel is either red, blue or green. The area around the My Little Pony's head (is the author a Brony?) is blown-up so you can see the Bayer pattern. The image is much too green because there are two green Bayer pixels for each red or blue pixel.



"Stars have different colours, with blue stars being hotter than red stars."

./rpi2dng image.jpg

This will output a file called **image.dng** that only contains the raw data. The file is missing important metadata that will be useful later on, but we can copy it over from the **.jpg** using another handy utility called *ExifTool*. First, install *ExifTool*, for example on Debian-based distros just do:

```
sudo apt-get install
libimage-exiftool-perl
```

And then copy across the Exif data with this:

```
exiftool -tagsFromFile image.jpg image.dng -o image.exif.dng
```

This copies the metadata from **image.jpg** and takes the raw data from **image.dng** and combines them in the output file **image.exif.dng**.

Using Rawpy

Rawpy is a Python module that provides tools for working with raw images. We'll also need *NumPy* for handling numbers and arrays, and *Matplotlib* so we can display images from Python. On a Debian-based distro you can get all of these with this:

```
sudo apt-get install python-numpy python-matplotlib libraw-dev
```

Open a text editor and enter the following lines:

```
import rawpy, matplotlib.pyplot as plt, numpy as np
raw=rawpy.imread('/home/foo/linuxvoice-imaging/image.exif.
dng')
print 'Sizes of the image:',raw.sizes
```

First we import the modules we need and set abbreviations called **plt** and **np** to save on typing later. The second line reads the data from the **dng** files into the raw object, and then prints out metadata on sizes.

Save these three lines in a file called **processraw.py**, and then in a terminal window **cd** into the directory where you saved it and run it like this:

python processraw.py

and you should see output showing the width and height of the raw image. For a Pi Camera the width will be 2592 and height will be 1944. Next, let's look at information about the camera's Bayer pixels. Add the following lines to **processraw.py** and run it again:

```
print 'Bayer pattern:\n',raw.raw_pattern
print 'Indices 0,1,2,3: ',raw.color_desc
```

The first two lines give this output:

```
Bayer pattern:
[[3 2]
 [0 1]]
Indices 0,1,2,3:  RGBG
```

This tells us that the Bayer pattern in this sensor is ordered so that in each group of 2x2 pixels, the two green pixels are top-left and bottom-right, the red pixel is bottom-left and the blue pixel is top-right. Pixel co-ordinates are such that (0,0) is at the top-left, and pixel (x,y) is x pixels to the right, and y pixels down. So, (0,0) will be green, and (1,0) will be blue, (0,1) will be red, and (1,1) green, and (2,2) will be green again, and so on. We can confirm a particular pixel's colour and extract its value as follows:

```
print 'Colour at 100,100:',raw.raw_color(101,100)
print 'Value at 100,100:',raw.raw_value(101,100)
```

which gives output

```
Colour of bayer pixel at 101,100: 0
Value of bayer pixel at 101,100: 32320
```

Note the arguments of these two methods are (y,x) and from the above we can see that 0 corresponds to red. Now let's attempt a graphical reconstruction of what the Bayer pixel array in the camera's sensor "saw". You can just add this code to the end of **processraw.py** and run it as before:

```
nx=raw.raw_image.shape[1]
ny=raw.raw_image.shape[0]
```

```
ris=raw.raw_image.astype(float)
rismax=ris.max()
rgb=np.zeros((ny,nx,3), 'float')
rgb[1::2,0::2,0]=ris[1::2,0::2]/rismax
rgb[0::2,0::2,1]=ris[0::2,0::2]/rismax
rgb[1::2,1::2,1]=ris[1::2,1::2]/rismax
rgb[0::2,1::2,2]=ris[0::2,1::2]/rismax
plt.imshow(rgb, interpolation='none')
plt.show()
```

There's a lot going on here so let's break it down. The first two lines store the width of the **raw_image** array in **nx** and height in **ny**. We need to give floating point values scaled between 0.0 and 1.0 to the plot the routine, so the next two lines prepare for this by turning the array into floats and setting **rismax** to the maximum value. *NumPy*'s zeros method is used to return a 3D array filled with zeros. You can think of this as having three layers indexed 0, 1 and 2, corresponding to red, green or blue respectively, where each layer has the same dimensions as the image.

The next four lines might look like a terrifying mess of colons, square brackets and numbers, but all they're doing is placing red Bayer pixel values into layer 0, green pixels into layer 1 and blue pixels into layer 2 of the **rgb** array at the same co-ordinates as they were in the original image.

To understand how this works, let's take a step back and look at 1D arrays in Python. Let's say we have an array **x=[1,2,3,4,5,6]**. Then **a=x[0::2]** says to start at index **0** in **x** and copy every second element to **a**, so **a** will be **[1,3,5]**. Likewise **b=x[1::2]** will contain only the even numbers from **x**. Now, if **z** is a six-element array filled with zeroes, then **z[0::2]=x[0::2]** will result in **z** being **[1,0,3,0,5,0]**. So, with this mind, we can translate the first line, ie **rgb[1::2,0::2,0]=ris[1::2,0::2]**, into English as "copy values from ris to rgb starting at (1,0) (which is a red pixel) and skipping 2 pixels in both the x and y directions".

The final two lines plot the data. The **interpolation='none'** setting ensures that the data is displayed as is and isn't smoothed to make it look more "attractive" when we zoom in. To see the Bayer pattern, you need to enlarge a small part of the image, which you can do by clicking the "Zoom to rectangle" button at the top of the window and then drawing a small rectangle somewhere on the image.

Build an image

We can now construct a more conventional image by using a simple method to demosaic the Bayer pattern. We'll take each group of four Bayer pixels and map them into one pixel in which the red and blue values

This image was taken using the piece of filter plastic that is shipped with the Pi NoIR camera. By comparing it with the colour balanced image above you can see that the leaves on the trees are much more obvious. This indicates that these trees are in rude health, as one might expect for a mature tree that's just sprouted fresh spring foliage.

are used unchanged, but the average is taken of the two green pixel values. We can do that with this code:

```
rgbi=np.zeros((ny/2,nx/2,3), 'float')
rgbi[::,::,0]=ris[1::2,0::2]/rismax
rgbi[::,::,1]=0.5*(ris[0::2,0::2]+ris[1::2,1::2])/rismax
rgbi[::,::,2]=ris[0::2,1::2]/rismax
```

Notice that there are no numbers around the double colons on the left-hand side now, because we intend the resulting image to be half the size of the original.

As you can see in the boxout, the image still doesn't look quite right, because no white balancing has been done, and this is especially noticeable if you compare it with the other image, which uses the Auto White Balancing (AWB) of the camera. We're not going to go any further into this subject here, but colour balancing can be done in Python, using clues from the camera's metadata. Have a look at **raw.daylight_whitebalance** and **raw.camera_whitebalance** and you'll see suggested coefficients for red, green and blue to achieve a decent white balance.

How green are leaves?

The hardware pixels used in cameras are sensitive to light that human eyes can't see that's just off the red end of the spectrum, known as the near-infrared. (Before you get too excited, it's the far-infrared that's used for night vision intensifiers.) Most cameras have a built-in filter to block this out because it will produce images that look quite unnatural to our eyes. However, the Pi NoIR (No InfraRed filter) camera is shipped without this filter. This makes it more suitable for some scientific tasks.

Plant leaves contain a substance called chlorophyll. Not only is it responsible for their green colour, but it is essential to photosynthesis, the process by which plants convert light to chemical energy. It turns out that a healthy, photosynthesising leaf will show a strong signature not just in the green, but also in the near-infrared part of the spectrum.

Where next?

If you're intrigued by raw image data but put off by the command line and coding you might want to experiment with some GUI tools. If you're already familiar with imaging-processing tools on Linux then the *UFRaw* plugin with *Gimp* or KDE's *Krita* will both enable you to work with raw data. Alternatively, have a look at *Darktable* or *RawTherapee* – both are specifically written for working with raw images.

If you're happy on the command line then you might like *DCRaw*, which lets you convert from many different raw formats to the common image formats, giving you control over aspects such as colour balancing and demosaicing through command line switches. In fact, *DCRaw* is the grandparent of *Rawpy* because *Rawpy* relies on *libraw*, which arose from a project to turn the source code of *DCRaw* into a library. If you wish to delve deeper into the complexities of raw camera data then there's plenty more to explore with Python and *Rawpy*, but if you're more comfortable with C++, you'll want to work directly with the *libraw* library.

Even if you don't need to work with raw data directly, an understanding of it not only gives insight into the tremendous power and flexibility of modern imaging devices, but also the nuances of how we perceive light and colour with our own eyes. LV

Andrew Conway absorbs infrared, predicts election results, watches the stars and uses Slackware Linux.

Demosaiced image

LEFT Simple demosaicing of the Bayer pixels. Each pixel in this image corresponds to a group of four Bayer pixels. The red and blue component of each pixel is set equal to the red and blue Bayer pixel values, and the green component is the average of the two green Bayer pixels. The colours are not natural because no white balancing has been performed.

RIGHT This image has been demosaiced and auto-white coloured balanced with default settings. It shows a much more natural range of colours, including the vivid red, green, yellow and blues of the children's toys in the foreground. The colours are still unnatural because the image was taken with a Pi NoIR camera, which is sensitive to the near-infrared which our eyes cannot see.

LINUX VOICE TUTORIAL

BATCH JOBS: AUTOMATE REPETITIVE TASKS

MIKE SAUNDERS

Need to do the same job on 1,000 different files? As always, the Linux command line comes to the rescue.

WHY DO THIS?

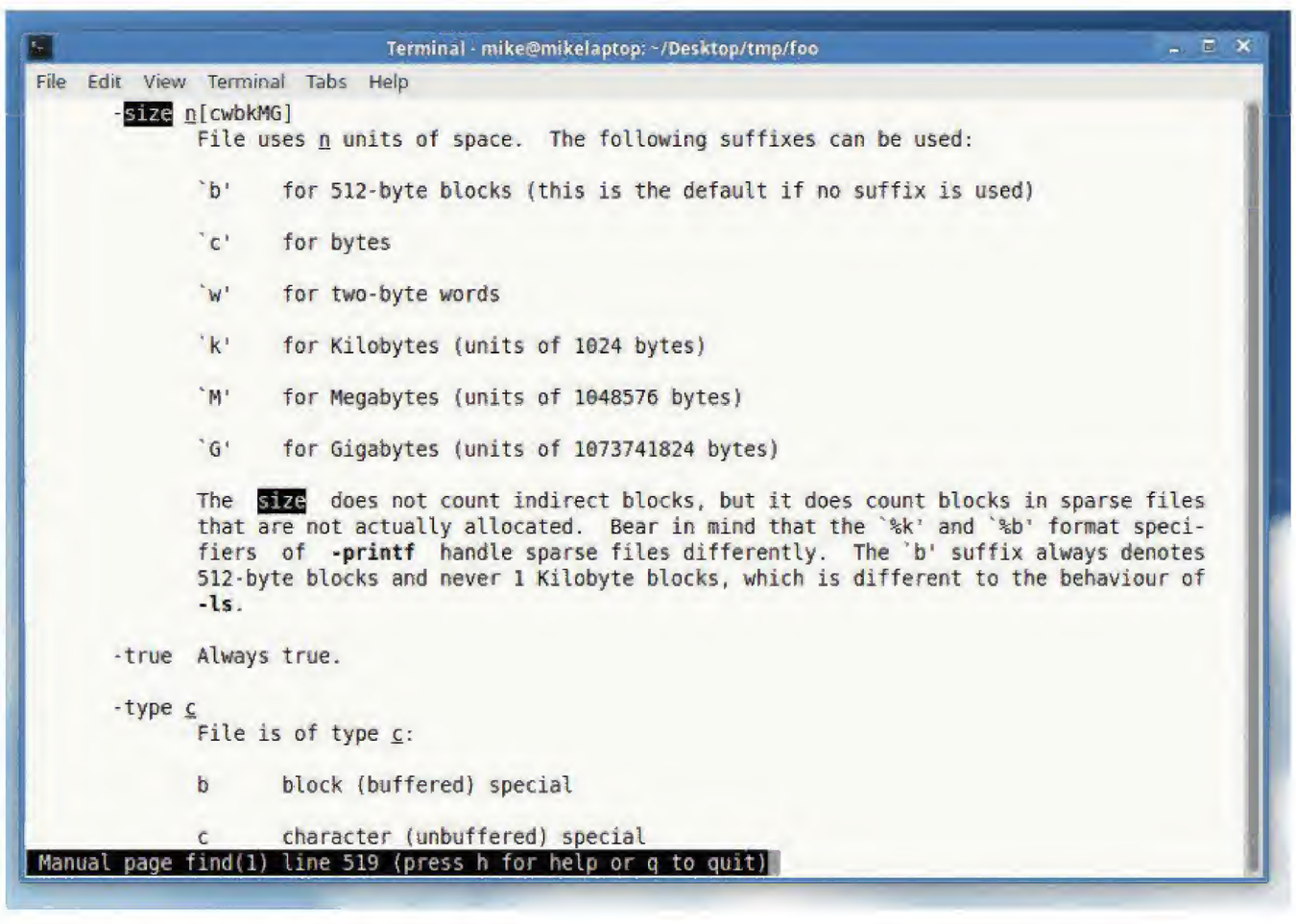
- Do hour-long jobs in seconds
- Automate your workflow
- Avoid getting RSI

New Linux users are often baffled by the command line interface (CLI). Why would anyone want to tap cryptic lines of code into a black box, when modern Linux distros provide enough point-and-click goodness for even the noobiest of noobs? Well, there are plenty of reasons to use the CLI, but our number one is this: it's fantastic for doing batch jobs. Whenever you need to perform the same task on hundreds (or even thousands) of files, nothing beats it. Sure, some graphical programs and file managers let you do batch jobs with a lot of fiddling around, but the CLI makes it very straightforward. So if you're new to Linux or you've never used the CLI to automate tasks before, open up a terminal and read on...

The key to performing a batch job is being able to iterate over a bunch of files. At the command line, the asterisk (*) character is a wildcard used to refer to all files, so if you enter a command like this:

```
file *
```

It will show information for every file in the current directory. Now let's say you have a bunch of JPG photo files from a recent trip called **austria_001.jpg**, **austria_002.jpg**, **austria_003.jpg** and so forth. On closer inspection of the EXIF data, you realise that the photos were taken just over the Bavarian border (it happens) and you want to rename them to **germany_001.jpg**, **germany_002.jpg** etc.



The **find** command has a huge array of options for narrowing down your searches.

You might be tempted to rename (move) them with something like this:

```
mv austria*.jpg germany*.jpg
```

But! This won't work at all. The asterisk wildcard is expanded by the command line shell before the **mv** command is executed, so it becomes:

```
mv austria_001.jpg austria_002.jpg austria_003.jpg
germany_001.jpg germany_002.jpg...
```

This makes no sense – you can't move multiple files to a single file (only into another directory). So we need a more canny way of doing this. Instead of trying to throw every filename at the **mv** command, we put **mv** inside a for-do-done loop, so that the command is executed for each file individually.

Consider this:

```
for x in *; do file $x; done
```

This does the same job as the **file *** command mentioned earlier, but instead of passing all filenames to the **file** command, it runs **file** on each one individually. This bit starts the loop:

```
for x in *;
```

It essentially says: for every file in the current directory (*), go through them one by one, and store the filename in the **x** variable – aka storage space – each time. (We choose **x** as a name here, but you can use something else.) Then we have the middle part of the loop, the bit that is executed for each file:

```
do file $x;
```

So, it runs **file** on the filename stored in the **x** variable. This happens for each file, and **done** signifies the end of the loop. Now, how do we modify this command to make it perform batch rename operations? Here we use text substitution, like this:

```
for x in *; do mv "${x}" "${x/austria/germany}"; done
```

Here, inside the loop we run an **mv** command to rename the files, and we **mv** it from the filename stored in the **x** variable to a modified version. The **x/austria/germany** part performs text substitution, taking the filename in **x** and replacing the part after the first slash with the part after the second. So in the end, this loop goes through all files and replaces all instances of **austria** with **germany**.

You can use variants of this command in many ways, eg for changing file extensions. You can also add text to the beginning of filenames like so:

```
for x in *; do mv $x 2015_$x; done
```

Here we add **2015_** to the beginning of each filename. Note that we use a simpler form of the **x** variable here – **$x** instead of **${x}** – because we're not

Handling unknown files

Here's a useful trick if you ever have a directory full of random filenames without extensions, or with incorrect extensions. This situation can occur if you use file retrieval software after deleting something, or in the case of corrupted filesystems. You can end up with thousands of files with names like NWI928AN – which isn't particularly useful.

As you've seen in the main text for this guide, you can use the **file** utility to determine the format of a file. This is a clever program that doesn't use file extensions to work out what a file is – after all, they can be changed to anything – but actually looks inside the file and searches for sequences of data. For instance, PNG files actually have the letters "PNG" in the first few bytes, so it's possible to identify them regardless of filename.

Run **file -i *** in a directory with some PNG files, and you'll see output like this:

```
somefile.png: image/png; charset=binary
```

This shows the MIME type for the file (**image/png**), so imagine we have a directory full of thousands of randomly named files, and we want to move all PNG images into a separate directory called **new**. We do this using the **mv** command as usual, and using backticks (as explored in the main text) we feed **mv** with a bunch of filenames generated by another command:

```
mv `file -i * | grep image/png | cut -f1 -d :` bar
```

Here we have a series of commands inside backticks (turning it into a larger command), and we use pipe characters to move the data between them. The **grep** part narrows down the list of files to just those with the MIME type of image/png, but we don't want that extra information in the list we send to **mv**. So we use the **cut** tool to take the first field of data (the filename), telling **cut** that fields are separated by colons (**-d :**). In the end, the command inside the backticks generates a list of filenames that have been identified by the **file** utility as PNGs.

This demonstrates how you can string together operations to generate the information that you need, and build up extremely powerful commands from a collection of small tools. This is the Unix way: standalone tools working together to make something big, instead of giant monolithic applications that try to do everything at once (and usually badly).

doing any text substitution, but merely adding some characters.

You can, of course, use these batch loops with other programs as well. One very popular command line suite for processing images is *ImageMagick*, which has a tool called **convert** that performs a wide range of operations on images. For instance, say you have a directory full of PNG files, and you want to create thumbnails from them with a maximum width of 100 pixels:

```
for x in *.png; do convert $x -resize 100 thumb-$x; done
```

Here, we run **convert** on all files ending with **.png**, create versions of maximum 100 pixels wide, and save them with a prefix of **thumb-**. The **convert** utility is capable of many other image editing operations, such as cropping, adding text and changing formats, so see the manual page (enter **man convert** at the command line) for more information.

Your time is precious

Sometimes it's useful to perform batch operations on files that match a certain date, and here's where we can use another cool trick at the command line. Backtick characters (`) enable us to perform a command inside another command, which sounds a bit strange at first, but look at this:

```
for x in `find . -type f -mtime -1`; do file $x; done
```

This runs the **file** command on every file that was modified in the last 24 hours, but nothing else. This is the important component of the above command:

```
`find . -type f -mtime -1`
```

Here we use the **find** utility to search for files in the current directory (**.**), and only normal files (**-type f**, so not directories or special system files), with a modification time of one day (**-mtime 1**). Because we surround this command in backticks, its output is passed back to the "**for x in...**" loop in the full command. There are other ways to pass output around, such as redirecting into text files, but this is a quick way to do it.

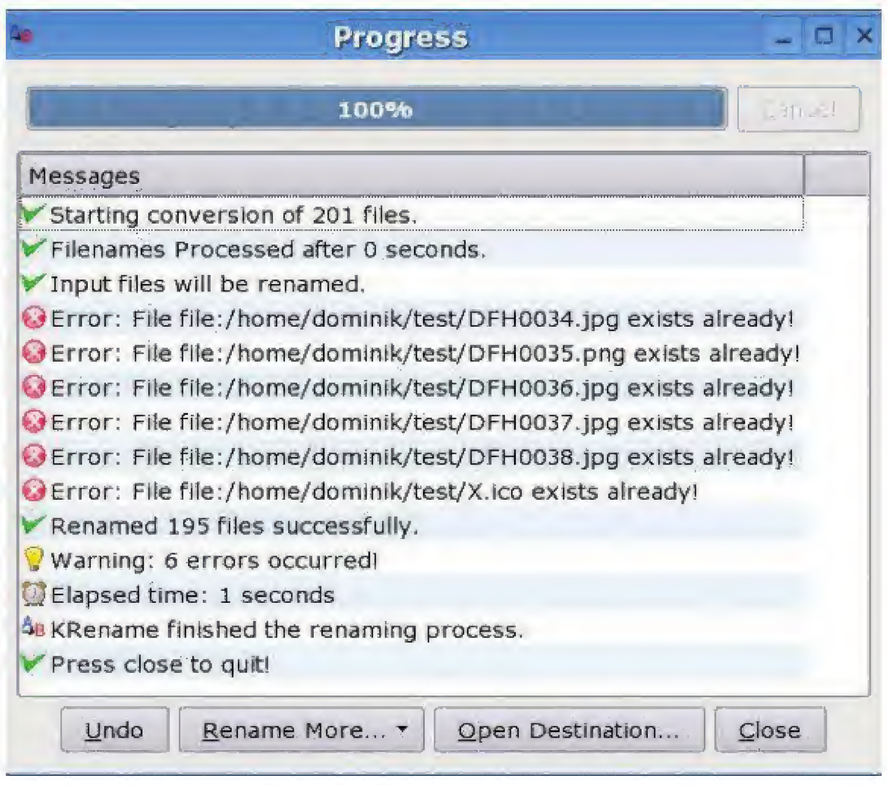
"Sometimes it's useful to perform batch operations on files that contain a certain date."

So, let's combine everything we've learnt into a beast of a command that will create 150-pixel JPG versions of all PNG files modified in the last three days. And to top it off, we'll restrict it to files that are at least 2MB in size (using the **-size +2M** option to find). Ready? Take a deep breath...

```
for x in `find *.png -type f -mtime -3 -size +2M`; do convert "${x}" -resize 150 "${x/.png/.jpg}"; done
```

Not bad, eh? With the **convert** command, we don't need to specify an option to say which file format we want to use – we just use a different extension for the output file. So we take the original filename and using text substitution, change **.png** to **.jpg** in the output file, and **convert** therefore knows which format we want to use.

So, now you know how to save heaps of time at the command line, and next time your Windows using friends or colleagues dismiss Linux because it involves lots of tapping stuff into a black box, you can laugh at how much time they waste with their pointy-clicky desktop fluff. LV



You can perform batch file rename jobs using graphical apps, but it's better to learn the command line approach to combine it with other tools.

Mike Saunders finds it fascinating that this "mouse" gizmo thing actually took off.

LINUX VOICE TUTORIAL

COBOL: THE LANGUAGE OF BUSINESS

JULIET KEMP

COBOL wasn't necessarily the best language, but as a tool aimed at non-specialists it was ahead of its time.

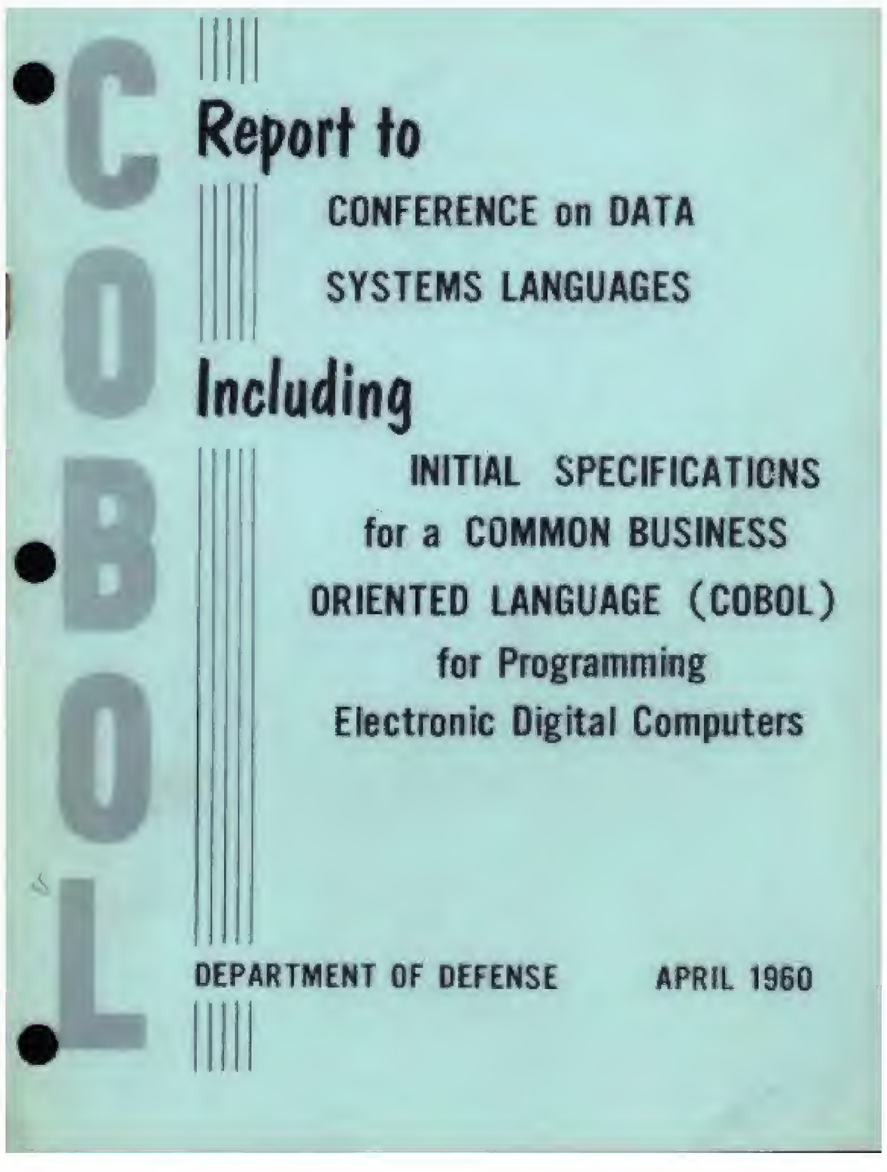
COBOL is the last of the big four late-1950s languages we've been looking at in this series. Like ALGOL, it was designed by a committee; but COBOL's main distinction was that it was aimed at businesses, and the design priority was to make it English-like and easy to understand. It's often seen as an outdated language, but there are billions of lines of COBOL still running on computers across the world, and the language is still being actively developed.

One of the problems computer users faced in the late 1950s was that programming was not only very expensive, but also non-transferable. Programs written for one computer couldn't in general be run on another computer, so your expensive program was limited to one machine. Programming languages that were semi-portable (like Fortran) were appearing, but these weren't aimed at business so much as at academics, and computer programming was of increasing interest to large businesses.

Mary Hawes, a programmer at Burroughs Corporation, was the first to call for something like COBOL. She wanted a cross-platform language that could run payroll, inventory, and other similar tasks: the sort of data processing that businesses wanted. Fortran simply wasn't set up to do this. Later in 1959, a group of people from various interested

As it turns out, the reports of COBOL's demise were greatly exaggerated. It's old, not dead.

organisations – academics, computer users, and computer manufacturers – met, and persuaded the US Department of Defense to sponsor the creation of a common business language. The DoD had a vested interest in the matter: owning over 200 computers already with another 175 on order, they were understandably keen to be able to run the same programs on all of them.



The cover of the first COBOL report, from 1960.

Design by committee

The initial meeting of the group, in May 1959, saw the participants describing a language that would be cross-platform; easy to use, maintain, and alter; English-based; and work across multiple environments. Various committees were set up, including a short-range committee which was to assess the currently available languages and begin to specify an interim new language. Grace Hopper was an advisor to this committee, and various computer manufacturers and government agencies were represented. Members included Jean Sammet (later creator of algebra system FORMAC), Betty Holberton (one of the first programmers of ENIAC and involved later in FORTRAN 77), and Bob Bermer (inventor of COMTRAN).

The committee was chiefly considering FLOW-MATIC, an English-based language invented by Grace Hopper and her team; AIMACO, a derivative of FLOW-MATIC; and IBM's COMTRAN, invented by Bob Bemer. Jean Sammet, remembering it later, described a certain amount of anti-IBM bias, and certainly

Y2K

You may remember, in the years before 2000, a lot of talk about the "Y2K problem". This boils down to the fact that, back in the 1960s and 1970s, when program space was at a premium, programmers chose to store dates as 6 figures: YYMMDD. So 1-Jan-1900 was 000101, and 1-Jan-1999 was 990101. The latter was larger than the former, and thus the second date was later than the first date. Even once space became less expensive, this continued to be the norm. Programmers simply weren't expecting the code they were writing in 1975 or even 1985 to last until 2000.

As it turned out, lots of this code was indeed still running come the 1990s. Which meant that 1-Jan-2000 would be stored as 000101 and the whole thing would (or might) break. The consequences of this would depend on the particular piece of code in question (is it comparing, or just displaying, or something else again?); which is to say, in any given case, no one knew without going to look. (And if you're going to look, you might as well crack on and fix it.)

This wasn't a COBOL-specific problem; it just happened that since COBOL was the most-used language in the 1960s and 1970s, most of the surviving code (ie the problem code) was COBOL. So, coming up to 2000, lots of COBOL programmers had many lucrative contracts to fix up all those 6-figure dates, either by changing them to 8-figure dates (ideal but expensive), or with a cheaper fix such as changing the 100-year 'window' to allow for the century change). This ran alongside a lot of media hysteria about planes falling out of the sky and so forth.

The date flipped round, nothing much happened, and there was a new round of media hysteria about how over-hyped the whole thing had been. On the other hand, you could see the fact that everything went smoothly as a credit to all those coders who put the effort in to make the changes in time.

(For another take on this, see **www.exit109.com/~ghealton/y2k/y2k_humor/Cobol.html**).

Grace Hopper seems to have pushed FLOW-MATIC's features quite heavily. Much later, in 1980, she claimed that COBOL was "95% FLOW-MATIC". However, while FLOW-MATIC's English language naming and many other features were included in the new COBOL spec, COMTRAN features were also included, such as some mathematical formulas, the picture clause (allowing data items to have both type and formatting defined), and a much better IF statement. What COBOL didn't have were functions with parameters, which would be criticised later.

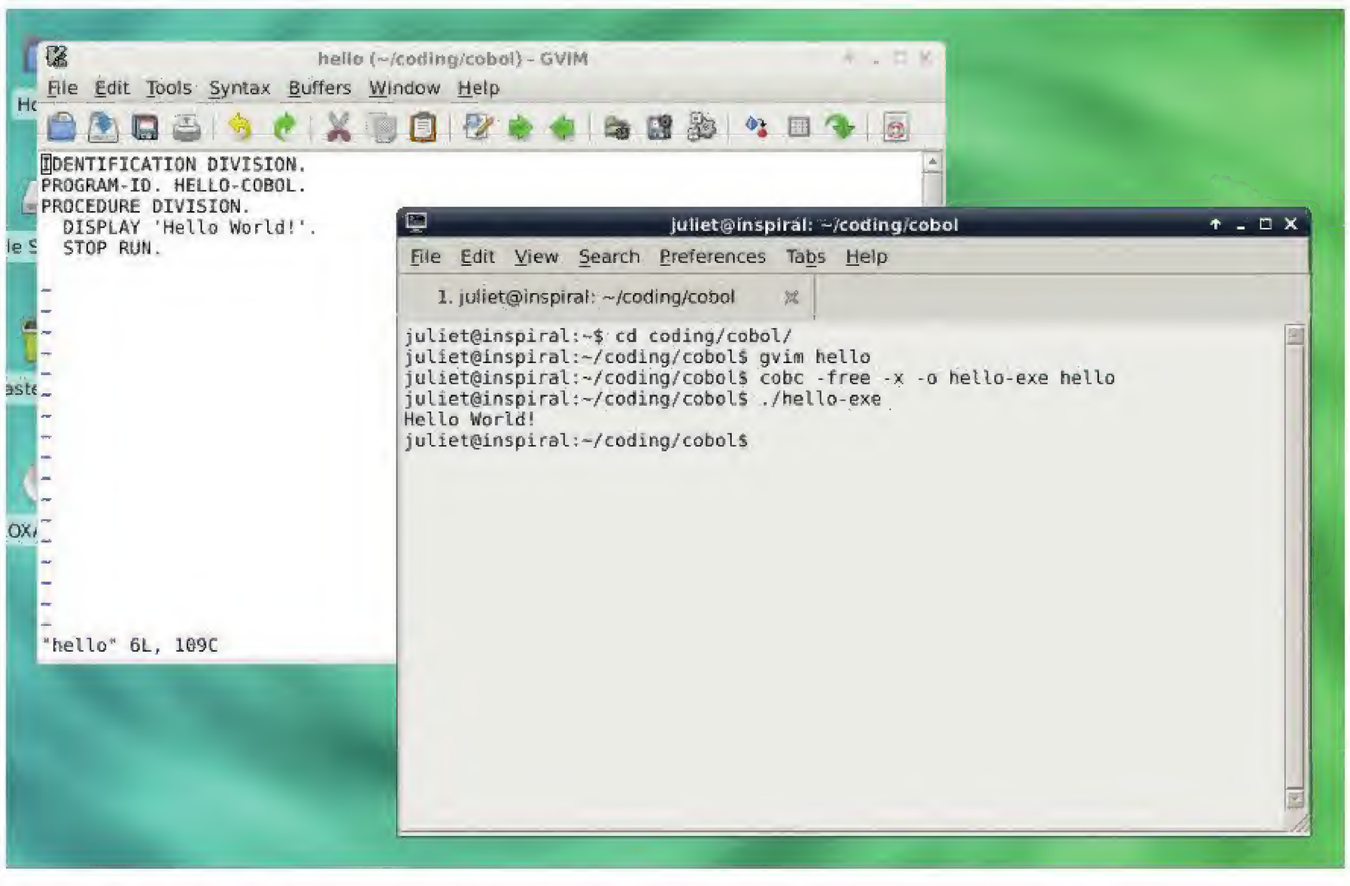
The size of the short-range committee made for slow progress; at one point Howard Bromberg bought a tombstone with "COBOL" engraved on it and sent it to the data systems director at the DoD. A further sub-committee was formed, which created the COBOL spec. COBOL 60 was signed off by the steering committee in January 1960, which meant that computer manufacturers could begin to create COBOL compilers. An RCA 501 was the first machine to successfully run a COBOL program, on 17 August 1960. The first cross-platform program was run in December, on a UNIVAC and another RCA machine. COBOL had achieved its aim: same program, different machines.

These first compilers were very slow, although by 1964, speeds were increasing. Various improvements were also made to the specification, in 1961, 1963, and 1965. (In particular, COBOL 60 included several logical flaws which were cleared up in the 1963 version.) By 1970, COBOL was the most-used programming language in the world.

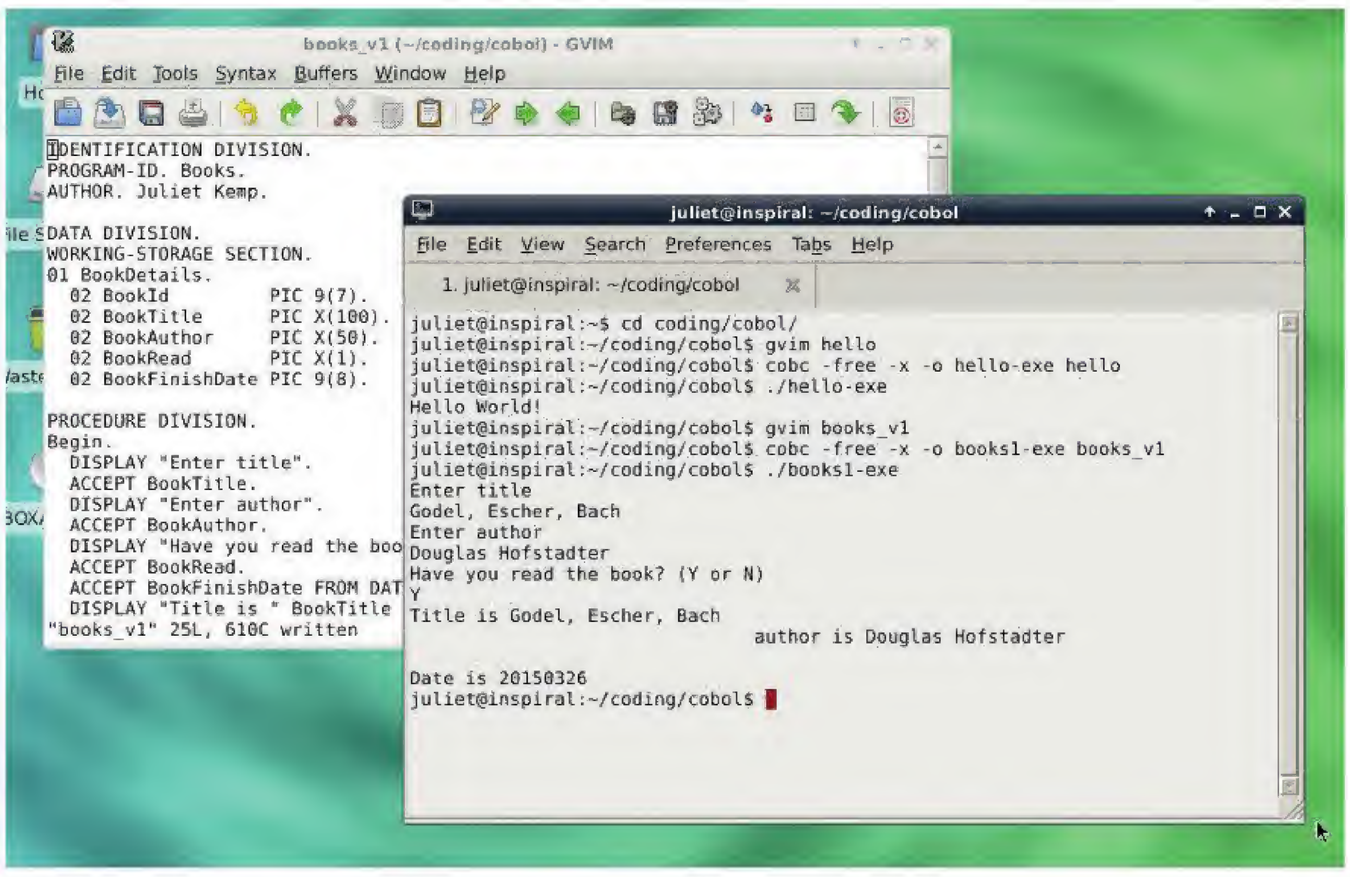
However, in the 1970s it came in for a certain amount of criticism as ideas of structured programming were developed. COBOL can be written in a structured way, but COBOL programs of the time often relied heavily on the GO TO statement. Attempts to rewrite code used the PERFORM statement (as used in the tutorial code below), but this didn't always work as clearly as it might. There was also no way to pass parameters into a procedure/function, which in retrospect looks like a very bad design decision. COBOL was also hard to modularise (in particular it was impossible to limit data access) at a time when modularisation was becoming popular. Further COBOL updates improved some of this, and in 2002 object-oriented programming was included in the spec.

Installation and Hello World

COBOL is still actively maintained, with COBOL 2014 the most recent release, and COBOL programs are still used globally across many different operating systems. Although some managers of these systems say they would like to migrate to another language, the bottom line is that migrating the billions of lines of actively-used COBOL is expensive, time-consuming, and risky. An expensive rewrite of an operational system purely to achieve the same thing in a different language is a hard sell, in budgetary terms, especially when the old code can instead just be migrated off aging mainframes onto modern kit. Not only that, but plenty of people would argue that the other mainstream languages simply don't replace the



It works! Our Hello World program in GnuCOBOL.



Writing to file – Note that odd spacing, which you can fix with the TRIM function.

business logic that COBOL does very well. COBOL is unlikely to be going anywhere in the near future.

The best bet for running COBOL on Linux is GNU Cobol, formerly known as Open COBOL and still often packaged under that name. Get it from **http://sourceforge.net/projects/open-cobol** or via your distro's package manager (in Debian the package is **open-cobol**).

Once you've installed it, here's a Hello World program to save as **hello**:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
PROCEDURE DIVISION.
 DISPLAY 'Hello World!'.
 STOP RUN.
```

Compile it with **cobc -free -x -o hello-exe hello**. The **-free** argument indicates you are using free source code format (see boxout). **-x** tells the compiler to build an executable program, and **-o NAME** saves the executable as the given name. Execute with **./hello**.

Entering and displaying information

Since COBOL is designed to handle data well, a database-type program seems like a good fit to try it out. We'll try a book database, storing book title, author, whether or not you've read the book, and date finished if so.

Here's a first iteration, which defines the data formats, asks for information, and prints it:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. Books.
AUTHOR. Juliet Kemp.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 BookDetails.
 02 BookId       PIC 9(7).
 02 BookTitle    PIC X(100).
 02 BookAuthor   PIC X(50).
 02 BookRead     PIC X(1).
 02 BookFinishDate PIC 9(8).

PROCEDURE DIVISION.
Begin.
 DISPLAY "Enter title".
 ACCEPT BookTitle.
 DISPLAY "Enter author".
 ACCEPT BookAuthor.
 DISPLAY "Have you read the book? (Y or N)".
 ACCEPT BookRead.
 ACCEPT BookFinishDate FROM DATE YYYYMMDD.
 DISPLAY "Title is " BookTitle " author is " BookAuthor.
 DISPLAY "Date is " BookFinishDate.
 STOP RUN.
```

COBOL programs have four divisions:

1. **Identification Division** Identifies the code, and in OO type COBOL, contains the class/interface definition.
2. **Environment Division** Contains configuration and input/output setup.
3. **Data Division** Sets up all program data, in multiple sections (files, working-storage, local-storage, linkage, report, and screen).
4. **Procedure Division** Contains the working code, in sections and paragraphs.

The **WORKING-STORAGE SECTION** of the **DATA DIVISION** contains any program-internal data; here, the fields that describe each book. ID will be auto-generated; title and author will need to be supplied for each book; as will whether the book has been read or not; and the date that it was finished on will also be stored. Note that each field has a PIC (picture) clause. This gives the type and length of the field. X stands for a character and 9 for a digit, and the length of the variable is given in brackets. So X(100) is a 100-character string.

The PROCEDURE DIVISION is the actual code of the program. DISPLAY displays text to the screen, and ACCEPT accepts user input and stores it in the given variable. ACCEPT name FROM DATE YYYYMMDD uses a built-in function to get today's date and store it in the specified variable, in the given format (eg 20150324 for 24 March 2015). We'll then display the stored data again before finishing the program with **STOP RUN.**. Note that all COBOL statements in any section of the program end with a full stop.

Compile and run with **cobc -free -x -o book-exe books; ./book-exe**. On running it, you'll see an unattractive space between the end of the book title and the "author is" output. This is because the book title is saved as a 100-character string, and COBOL automatically pads the string to that length with spaces. To avoid outputting those spaces, you can use OpenCOBOL's **TRIM** function:

```
 DISPLAY "Title is " FUNCTION TRIM(BookTitle) " author is "
FUNCTION TRIM(BookAuthor).
```

The code so far just takes a single entry and outputs it; it doesn't store it. Let's write it to a file:

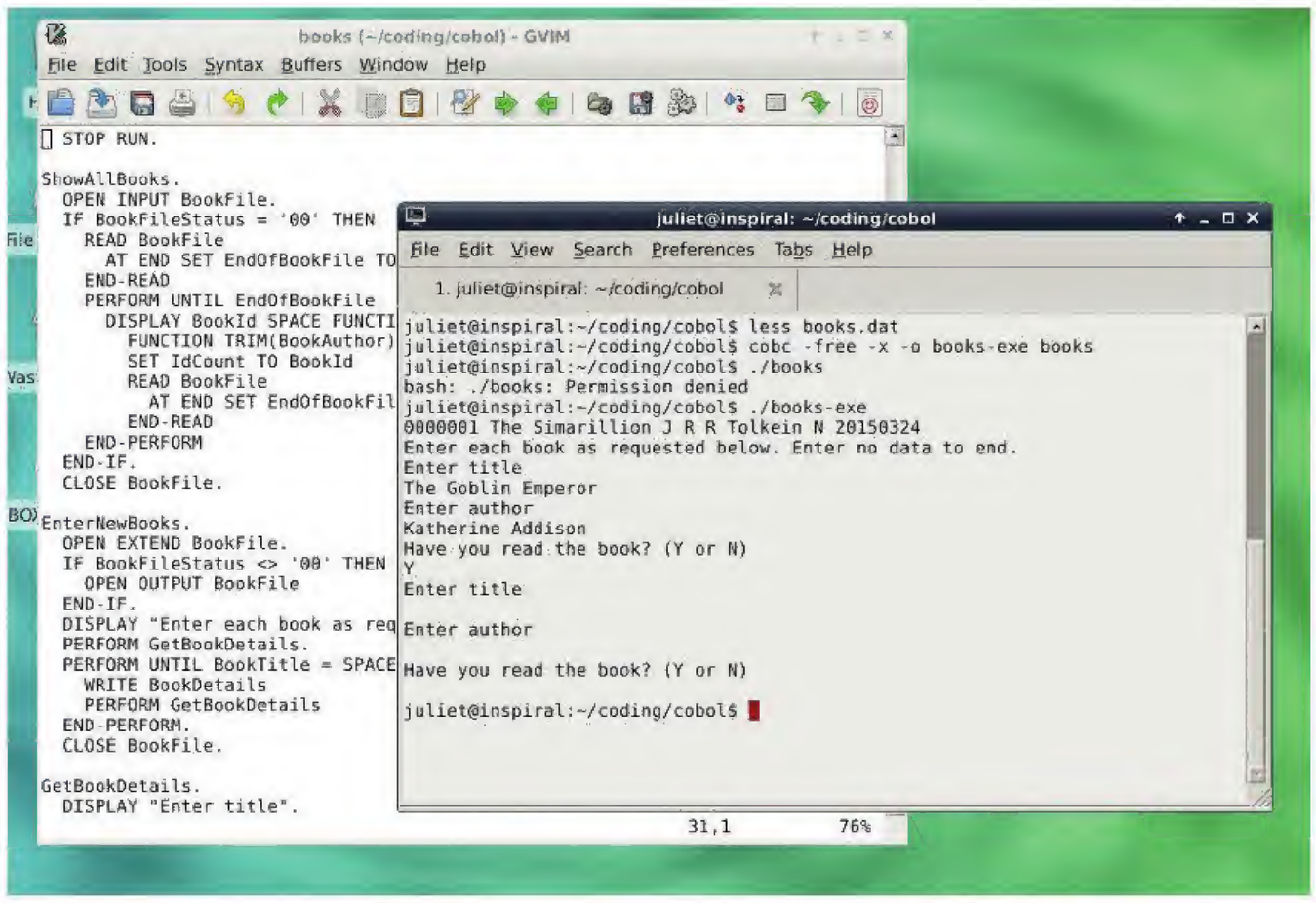
```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
 SELECT BookFile ASSIGN TO "books.dat"
```

```
ORGANIZATION IS LINE SEQUENTIAL.

DATA DIVISION.
FILE SECTION.
FD BookFile.
01 BookDetails.
 * remaining sections of BookDetails as above

WORKING-STORAGE SECTION.
01 IdCount PIC 9(5) GLOBAL.

PROCEDURE DIVISION.
Begin.
 OPEN OUTPUT BookFile.
 DISPLAY "Enter each book as requested below. Enter no data to
end.".
 PERFORM GetBookDetails.
 PERFORM UNTIL BookTitle = SPACES
   WRITE BookDetails
   PERFORM GetBookDetails
 END-PERFORM.
 CLOSE BookFile.
 STOP RUN.
GetBookDetails.
 DISPLAY "Enter title".
 * get title, whether or not read, and finish date as before
 ADD 1 TO IdCount.
 SET BookId TO IdCount.
```

This time there is an **INPUT-OUTPUT SECTION**, which controls the files used. The label **BookFile** is assigned to a specific filename, and the file organisation is line sequential, which means that it must be read line by line in the given order. (Files can also be indexed, which allows random access by key.)

The data fields are the same, but under the **FILE SECTION** instead of the **WORKING-STORAGE SECTION**, which just contains a global variable to act as a counter. The **FILE SECTION** describes the records of any files used by the program, whereas the **WORKING-STORAGE SECTION** defines program-internal variables, which will not be written out to a file.

The file is opened for output, and the initial user information given. Then we use the **PERFORM** keyword to call the **GetBookDetails** paragraph. **PERFORM UNTIL** provides a loop in which the program continues to ask for input until it gets an empty title field. **WRITE** writes the information



Outputting and entering information. You need to hit return through all three fields to finish, which is something else that could be improved.

to file, then **PERFORM** again asks for a new set of information. Note that the first **PERFORM** call is outside the loop, so the first **WRITE** call has something to write. This avoids writing a blank line at the end of the file. Note too that there is no full stop until after **END-PERFORM**; **PERFORM...END-PERFORM** is a single statement. Finally, remember to close the file at the end before stopping the program.

In COBOL, a paragraph is a block of code, of one or more sentences, labelled with either a language-defined or a programmer-defined name. It continues until the next section or paragraph is encountered (or the end of the code). So **FILE-CONTROL.** above labels a paragraph, and so does **GetBookDetails.**, which comes after the main body of the program. This paragraph mostly does the same as in the first version of the code. However we also use the global counter to set the **BookId** field. (Add one to it first to start at one.) In effect this is a lot like a function, but unlike functions in most other languages, you can't pass parameters into it; instead it just has access to the global program variables.

If you compile this and run it a couple of times, you'll see that as stands, it overwrites the **books.dat** file each time. You can use **OPEN EXTEND BookFile** to add to the end of the file, but this will fail if the file doesn't already exist. It isn't perfect; we'll leave it in as an exercise for the reader to see if you can fix this.

A great resource to find more code to look at and experiment with is the University of Limerick's sample programs (**www.csis.ul.ie/cobol/examples**), which were a great help to us when writing the tutorial code. There are also useful links at the University of Michigan COBOL page (**http://groups.engin.umd.umich.edu/CIS/course.des/cis400/cobol/cobol.html**). It may have a slightly elderly feel to it, but COBOL is still an interesting, robust, and surprisingly common language. LV

Juliet Kemp is a scary polymath, and is the author of Apress's *Linux System Administration Recipes*.

Free vs fixed format

As with other languages of a similar age, COBOL originally had a fixed format structure, in which source code was written in lines of 72 characters, consisting of a sequence number, an indicator area, area A (in which section or paragraph names begin), and area B (in which other code sentences begin; so code is indented). By default, OpenCOBOL still expects this format.

However, modern COBOL (since 2002) also accepts, with the **-free** flag, free-format source code, which does not have the same limitations. The code here all uses free format.

LINUX VOICE TUTORIAL

GET STARTED WITH GNOME BUILDER

GRAHAM MORRISON

We take a look at a new, clean and particularly awesome integrated development environment for Gnome.

WHY DO THIS?

- Install a cutting edge IDE with a wonderful UI
- Create your own GTK+ and Gnome apps

Many people consider Gnome the closest thing to a default desktop for Linux, but Gnome and its toolkit, *GTK*, have never had a great set of developer tools, nor a modern integrated developer environment (unless you include *Vim* and *Emacs*). *Glade* is still an excellent tool for user interface design, but it doesn't help developers take the next step. *Anjuta* isn't bad either, and it does integrate well with *Glade*. But *Anjuta* hasn't been developed for a while and can be intimidating.

Meanwhile, *Qt* and KDE developers get both *KDevelop* and the wonderful *Qt Creator*, and there's *Eclipse* for everyone else, although we admit that *Eclipse* would easily win any competition for the most intimidating IDE. But with Gnome's current focus on usability and user experience, it seems fitting that there should be a better option, and one that's more in tune with Gnome's new principled user interface. And now there is – *Gnome Builder*. Let's take a look at the future of Gnome development...

Step by step: Manage your projects with Builder

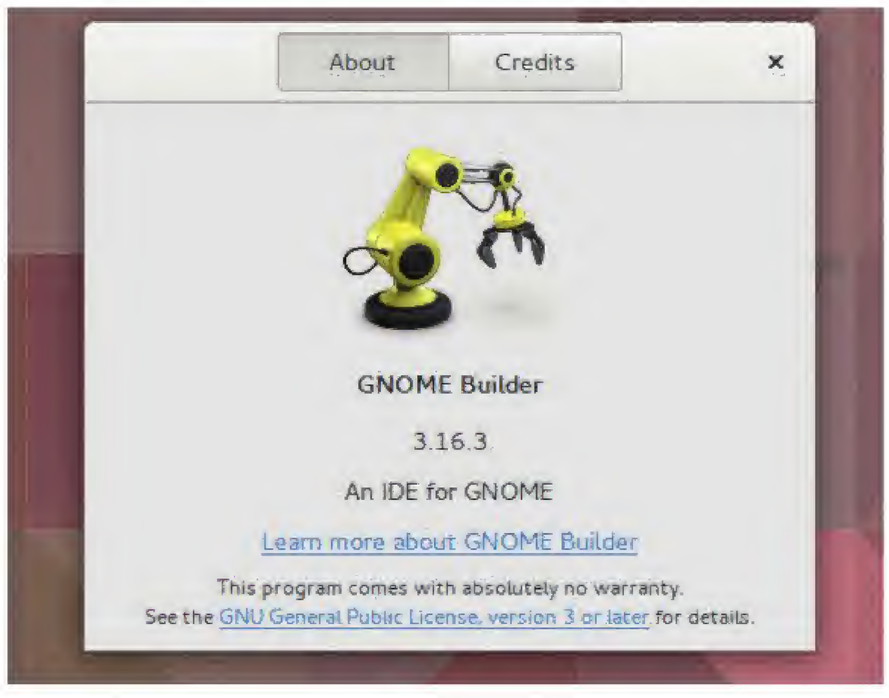
1 Installation

Gnome Builder is the brainchild of Christian Hergert, and the product of a phenomenally successful Indiegogo crowdfunding campaign. Launched in December 2014, the campaign asked for $40,000 to enable Christian to work full time on the project, after he'd already quit his job and put four months into the prototype. The campaign reached its target by raising $55,204, and the rate of development since the campaign's conclusion has been incredible – so much so that *Gnome Builder* has had several releases and is already functional. It's got a few rough edges and many features yet to be implemented, but it's great fun to work with, especially with its inspired design. It's the perfect excuse to try a bit of Gnome development, which is exactly what Christian wanted to achieve. Installation can be tricky if you don't want to compile it yourself, which is fair enough for an application designed for developers. The simplest solution is to use the shiny new Fedora 22, where *Builder* is already in the repositories.

2 Open a project

For non-Fedora users, there's also a PPA for Ubuntu and *Gnome Builder* is easily installed through Arch. We installed version 3.16.3, and while *Builder*'s user interface is already well thought out, it's also being rapidly developed, so it's possible that features may have changed slightly by the time you read this. The best way to get a taste of what *Builder* can already do is to use its text editor because we think it's already one of the best we've seen.

Builder is a strict adherent to Gnome's minimalism, which you'll notice when you launch the application for the first time. Before you've even said hello it lists any automake projects it finds within your home folders, and you can simply click on one of these to open a project. Alternatively, you can click on 'New' in the top-left. Here you get to select an existing folder, empty or not, and call that a new project, or enter the Git URL of a project you'd like to download. We started with an empty folder, which will then give you an empty editor view on the right.



3 User-interface tour

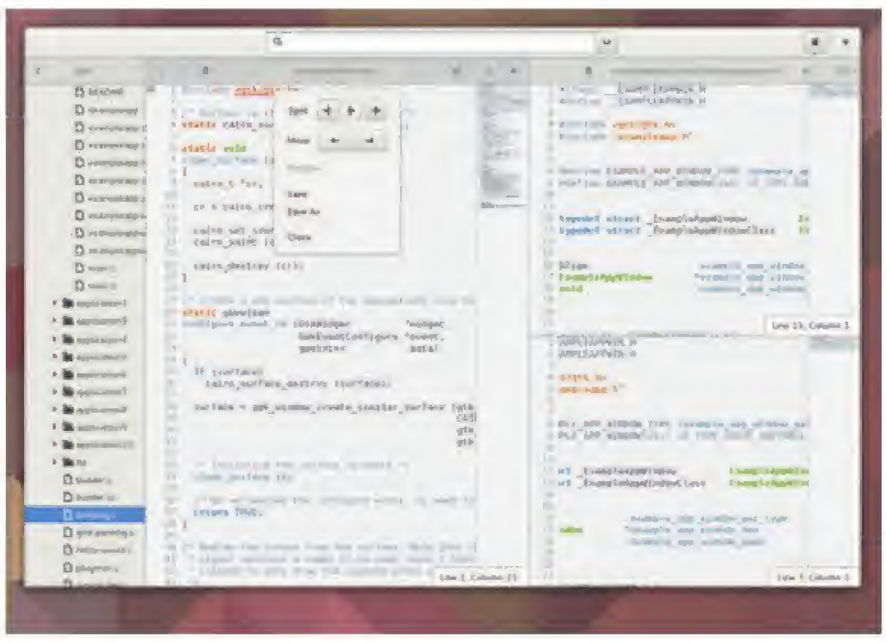
There are three ways to configure what you see. Click on the top-right menu, and the View option listed enables and disables the left-hand panel, which is used to navigate files of a project folder. Beneath this, labelled 'Plain Text' by default, is a button that enables you to specify the type of the file you're editing.

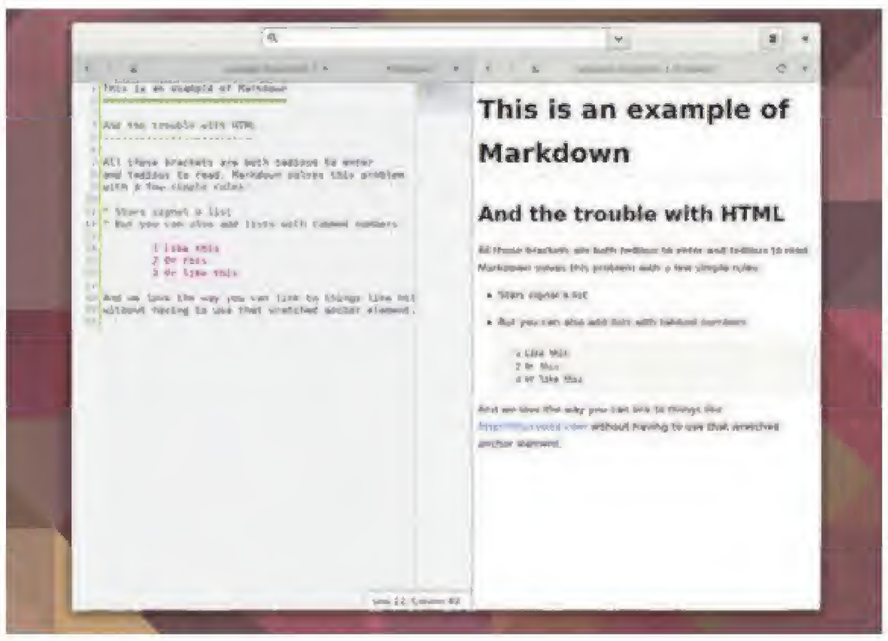
There are more options here too, for adding things like line numbers or the excellent auto-indentation. Finally, our favourite features are hidden beneath the filename of the file you're editing, just above the editing window. Click on this and another small panel appears. The top row of icons lets you split the view horizontally or vertically, allowing you to have more than one file open at a time, or the same file but different locations. You can use the arrows beneath to move between open files.

4 Get editing!

Another of *Builder*'s nicest features is a built-in real time preview for HTML and markdown documents – the latter being especially impressive, as there isn't a decent native open source Markdown editor for Linux we know of. Markdown is an incredibly useful way to add formatting information to pure text files by using a variety of simple and intuitive symbols that don't get in the way of the legibility of the original text.

To see the preview in action, start a document like this in the editor. If *Builder* doesn't automatically change the file type, make sure you change it to read 'Markdown', or HTML if you'd rather preview that. Now click on the filename to open the split window view and add a vertical or horizontal split. Select the panel you want to be the preview and use the same menu to enable the 'Preview' option.



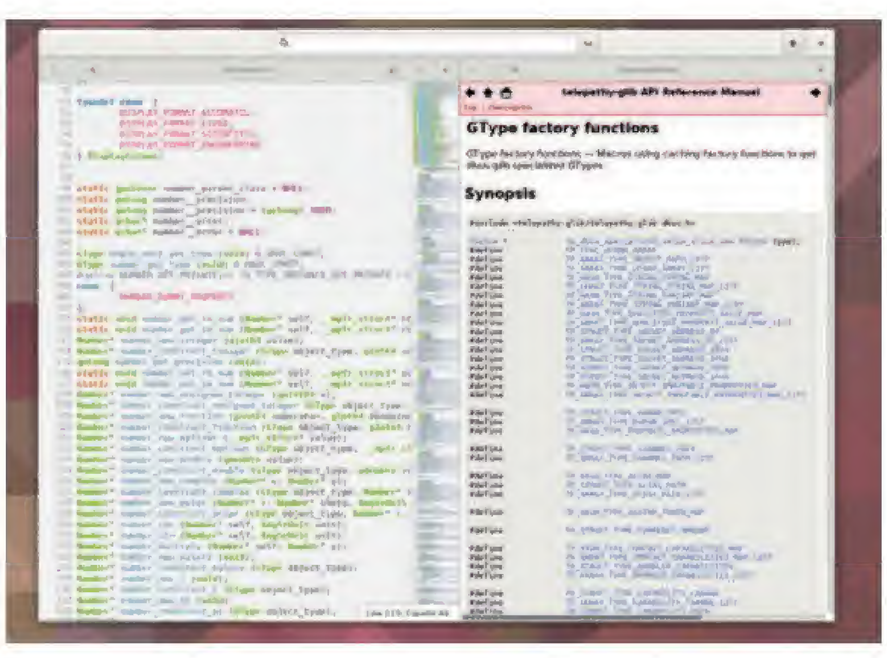


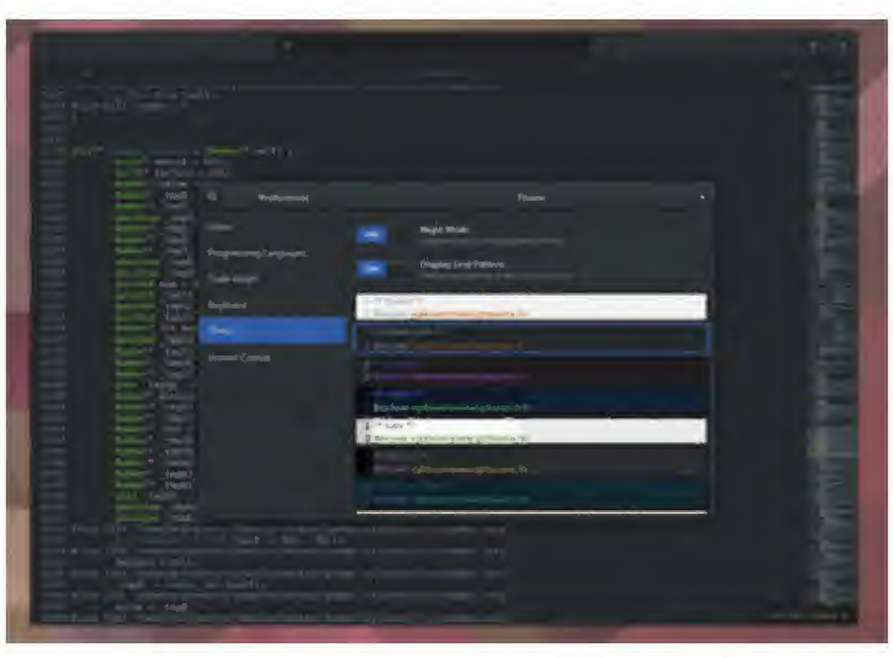
5 Get coding!

The easiest way to get a build environment working and start playing is to download a simple Gnome project with an environment in place. We went with *gnome-calculator*, which can be grabbed as a tarball from **download.gnome.org**. From a command line in the untarred calculator folder, type **./configure** to generate the autotools scripts. You can then type **make** to build the project, or use the side panel in *Builder*. You can now take advantage of all the features currently implemented, including error highlighting, auto-completion for C and C++ and the global search. This will list everything related to the project and let you skip between search hits within your files, but it also links to API references which will appear in a new panel to the right of your code.

6 Change the preferences

You can press F4 to switch between the source and the header file, but there are more options in the preferences panel if you need to change the keyboard layout, including both *Emacs* and *Vim* emulation. There are options for all of the programming languages that the editor supports. Each can have its own margins and indentation. But our favourite feature is the night mode, which can be enabled from the 'Theme' page of the preferences panel. The Builder Dark theme is perfect for late-night coding sessions, which are something *Builder* has inspired us to look into. *Builder* may still be at only an early stage of development, but the small team has already produced an excellent IDE with a lovely, minimal and deceptively comprehensive user interface. LV





LINUX VOICE TUTORIAL

JAVASCRIPT: CREATE A SECURE, ANONYMOUS CHATROOM

BEN EVERARD

Build a fully functional web app with enterprise-level security using just one language.

WHY DO THIS?

- Get started with Node.js, the trendiest web tech of 2015
- Learn to use web sockets for two-way communication in a web browser
- Develop a secure messaging service and keep your chats private

This month we're looking at privacy online, and to help with this we're going to create a web chatroom with end-to-end encryption and anonymity. To keep things simple, we'll do everything in a single language: JavaScript. This language has traditionally been used to add small functions to websites, but it's grown considerably and is becoming popular on the server thanks to Node.js.

Node.js is an event-driven non-blocking platform. This makes it particularly suitable for interactive web apps that have two-way communication between the browser and the server. There are two bits of software you'll need to install before you can get started, Node.js itself and *npm* (the *Node Package Manager*) for installing the additional modules we'll need. You should find both in your distro's repositories, though make sure that you're installing the right node software as there's also a radio application called *Node*. On Ubuntu-based distros, Node.js is in a package called **nodejs**, and you can install everything you need with:

```
sudo apt-get install nodejs npm
```

Node is based on the JavaScript engine from the Chrome browser, so the basic structure of the language is identical. However, it also comes with additional features to let it do things like run a web server and read the filesystem.

Node.js does come with a web server that you can use without any additional modules, but to make everything a little cleaner, we're going to use the Express.js framework. You'll need to install that with:

```
sudo npm install express --save
```

As is tradition, we'll start our journey into Node.js with 'Hello World'. To do this, we'll need to start up a server in Node.js that serves a single page which just contains the words 'Hello World'.

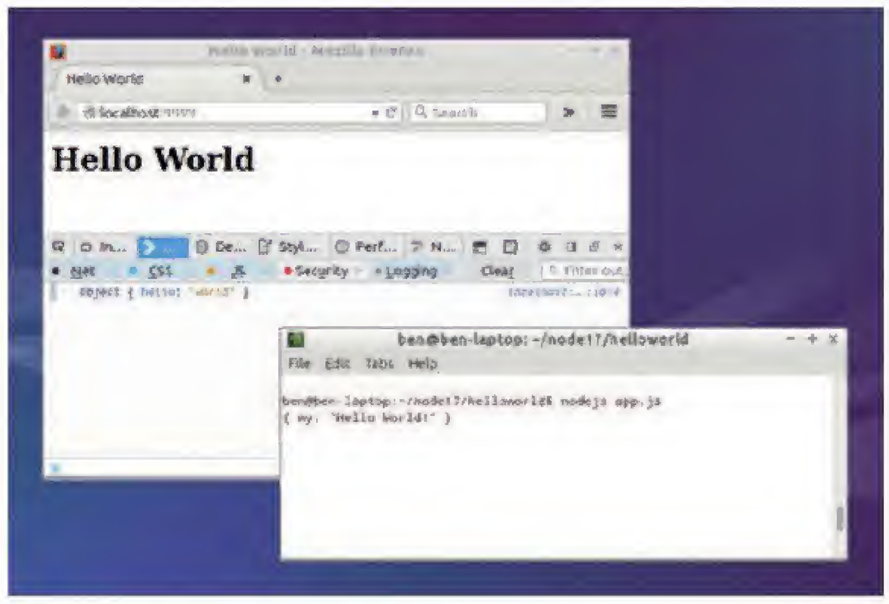
The Node.js code for this is in a file called **app.js**:

```
var app = require('express')();
var server = app.listen(9999);
app.get('/', function (req, res) {
 res.sendFile(__dirname + '/index.html');
});
```

The first line imports and initialises the Express.js framework. The second line creates a server that's listening on port 9999 (you can use a different port if you prefer, but our test machine is running other servers on the normal web ports).

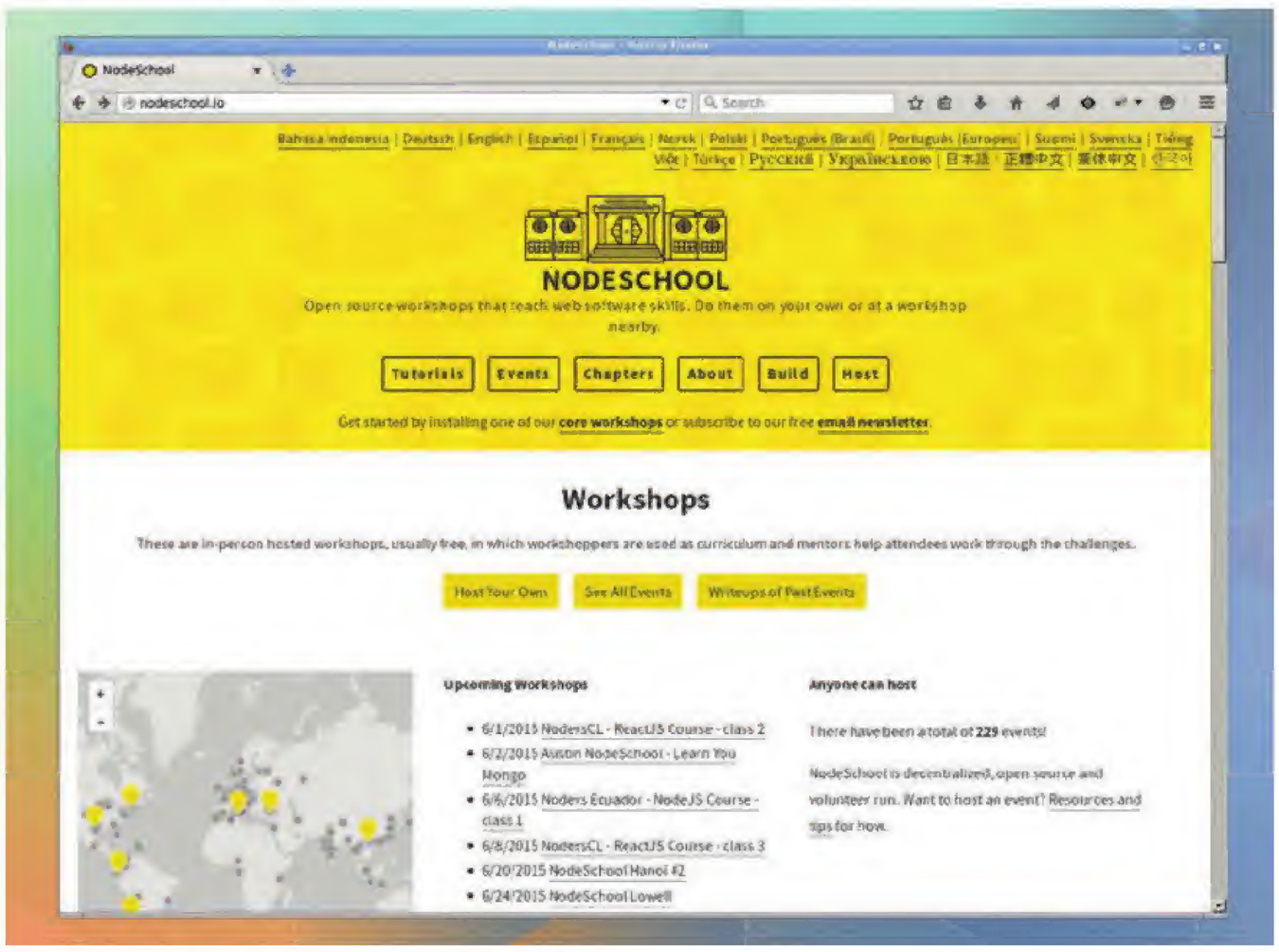
In Express.js, you have to tell the server what to return for each path using **app.get** (there's also **app.post**, but we won't be using that in this tutorial). **app.get()** takes two parameters: the first is the path that the browser requests; the second is the function that is used to process this request. As you can see, this is an anonymous function that itself takes two parameters: the request and the response. The request can be used to get more information about the HTTP request, and the response is used to formulate what is sent back to the browser. In this case, we just send the file **index.html** that's located in the same directory that Node is being run from.

JavaScript code often uses functions passed as parameters like this, known as callbacks. When they're used well, it can make it easy to write event-



A triple hello world using Node.js, Express.js and Socket.IO all in a single web app.

If you want to learn more about Node.js, NodeSchool has online tutorials, and real-world workshops to help you get started: **http://nodeschool.io**.



driven code. When they're used badly, they can lead to unreadable spaghetti code.

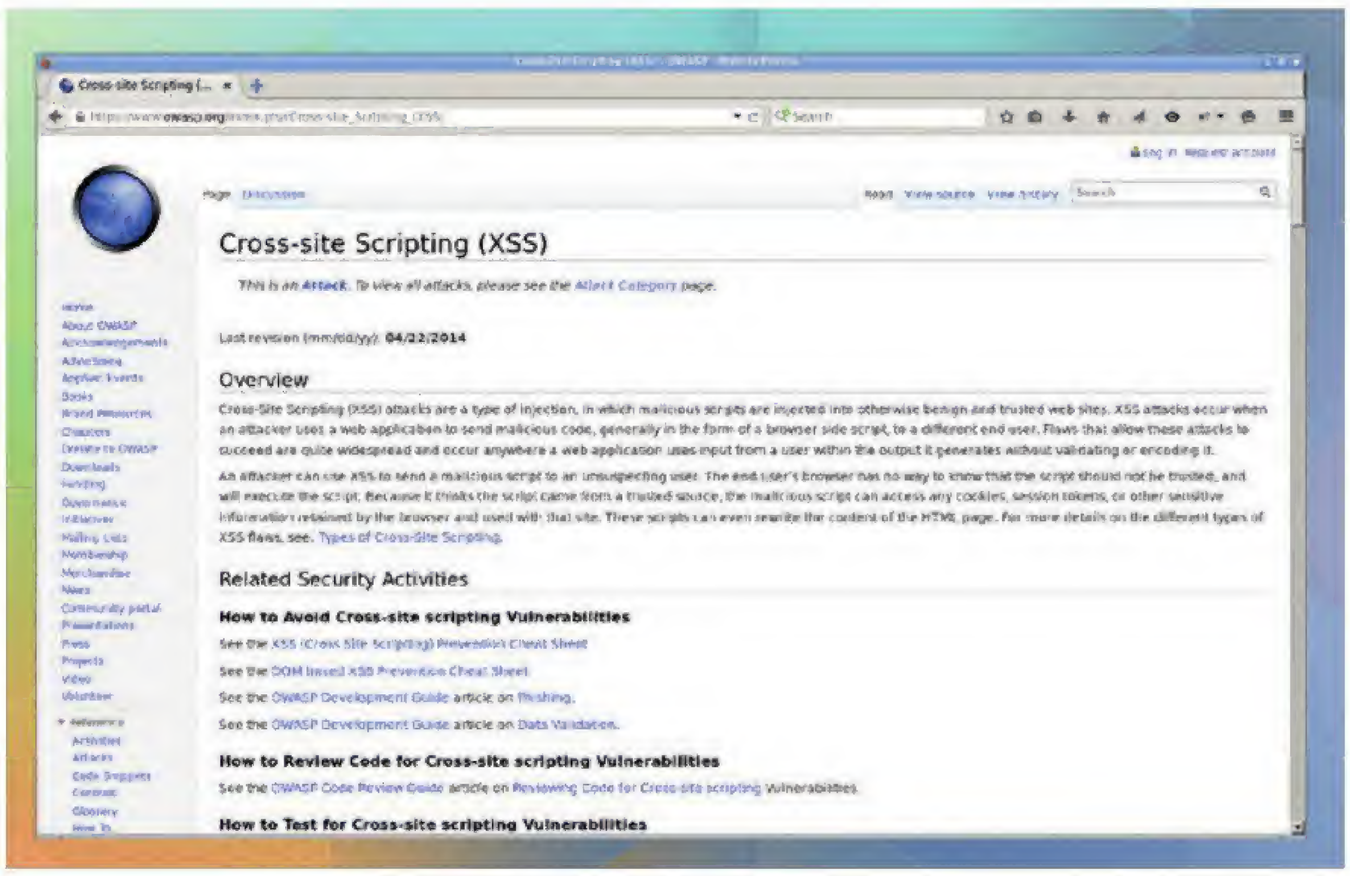
That's the Node.js code sorted. We just need the HTML file called **index.html**, which in this case is really simple.

```
<html>
<head>
<title>Hello World</title>
</head>
<body>
<h1>Hello World!</h1>
</body>
```

With those two files created, you can start Node with:

```
nodejs app.js
```

Then, if you point your web browser to **http://localhost:9999**, you'll see the Hello World web page.



The Open Source Web Applications Security Project (OWASP) has some resources to get you find and stop XSS attacks: **www.owasp.org/index.php/Cross-site_Scripting_(XSS)**.

Going both ways

Now, let's move on to our chat application. The web has a client–server model where a client (a web browser) requests some data (a web page) from a server. That content is sent, and the connection finishes. For a chat session, though, we need to keep a line of communication open between the browser and the server, and have some way of pushing data back and forth between the two in a manner that's more like peer-peer software than client–server software. For this we'll use Socket.IO, a JavaScript framework with components for both the browser and the server.

First, you need to install Socket.IO for Node.js with:

```
sudo npm install socket.io
```

Now we need to add both the server and the browser parts of Socket.IO to our Hello World app. First the server part. Change **app.js** to:

```
var app = require('express')();
var server = app.listen(9999);
var io = require('socket.io')(server);
app.get('/', function (req, res) {
 res.sendFile(__dirname + '/index.html');
});
io.on('connection', function (socket) {
 socket.emit('news', { hello: 'world' });
 socket.on('reply', function (data) {
            console.log(data);
 });
});
```

Socket.IO, like Express, uses callbacks to set functions that should run on certain events. The **io.on** line **('connection', ...** is used to set callbacks that happen to every single browser that connects to the server. In this case, the line:

```
io.on('connection', function (socket) {
```

creates an anonymous callback function that runs every time a new client connects to the server. When a client connects, it creates a new socket, and it's this socket object that's passed as a parameter to this callback function.

The callback function does two things. First, it sends (or emits) a chunk of data that will trigger a 'news' event in the connected client (more on that later), and second, it creates another callback event specific to this socket. In this callback, every time a reply event is triggered, it logs the output. This logged data will appear in the terminal where Node.js is running.

The code in **index.html** should be changed to:

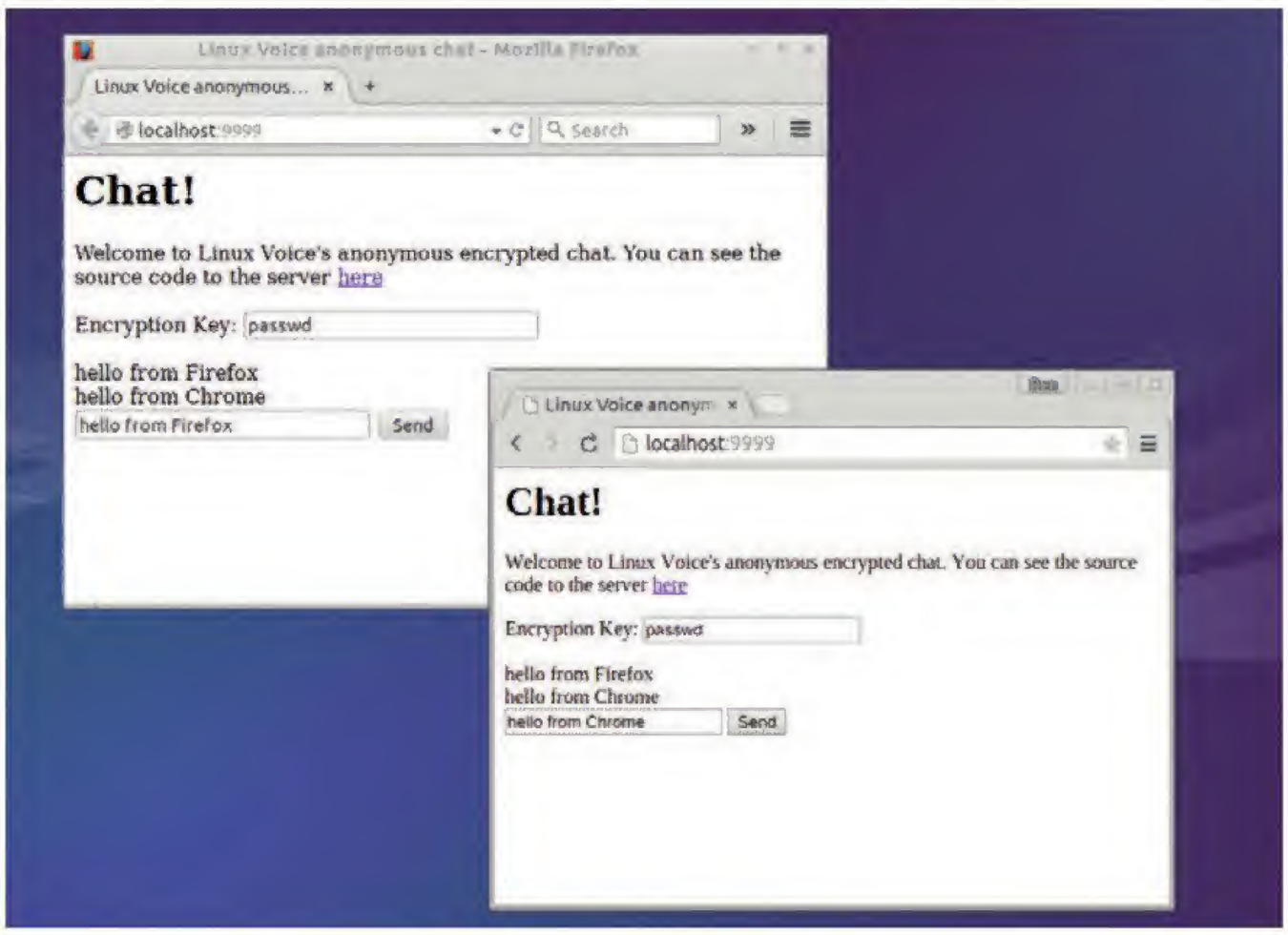
```
<html>
<head>
<title>Hello World</title>
<script src="/socket.io/socket.io.js"></script>
<script>
 var socket = io.connect('http://localhost:9999');
 socket.on('news', function (data) {
            console.log(data);
            socket.emit('reply', { my: 'Hello World!' });
 });
</script>
</head>
<body>
<h1>Hello World</h1>
</body>
```

This loads the Socket.IO JavaScript library from the server (this path is automatically set up when you initialise Socket.IO in Node.js). This connects to the server, and creates a callback for the news event. This is the news event that's triggered by the **socket.emit** line in our code. In the server code, we sent some JSON data (**{hello: 'world'}**). This is the data that's the parameter of the callback function. The callback sends this data to the console log, and then calls **emit** to trigger a reply even in the socket on the server.

The console log in the browser can be viewed in the developer tools.You can enable these in *Firefox* or *Chrome* by pressing **Ctrl+Shift+I**.

You can run this code exactly as before. You'll need to end (with Ctrl+C) and re-run **nodejs app.js**, and then reload **http://localhost:9999** in your browser. This time, though, you'll get triple Hello World. You should see the greeting in the main browser window, in the browser console and in the terminal where you're running Node.js.s

We now have all the pieces we need to build our



The final app sending encrypted messages between *Chrome* and *Firefox*. Take that NSA!

chat application. There's a method for serving the HTML page and a method for sending data back and forth between the server and the browser.

Pulling it together

The server side of our chat program is really simple – all we need to do is receive data from a client, and then send it out to all clients. This is done by replacing the **io.on** connection callback with the following:

```
io.on('connection', function (socket) {
 socket.on('news', function (data) {
          io.emit('news', data);
 });
});
```

The browser code is a little more complex. This has to accept user input to forward to the server, and receive other users' chat data and display it on the screen. We'll look at this in two parts. First, the HTML for the web page **<body>** section is used for the user interface. This should be:

```
<body>
<h1>Chat!</h1>
<p>Welcome to Linux Voice's anonymous encrypted chat. </p>
<div id="chat">
</div>
<input id="msg" type="text">
<input type="button" value="Send" onClick="send();">
```

This has a ** ** with the ID chat. We'll use this to display the messages that we get from the server. There are also two inputs: a text field and a button. The text field is where the user enters their message, and the button runs the send function when it's clicked to send this message to the server.

The **<head>** section of the page then contains the JavaScript code to control these (nb this comes before the body section in the **index.html** file):

```
<head>
<title>Linux Voice anonymous chat</title>
<script src="/socket.io/socket.io.js"></script>
<script>
var socket = io.connect('http://localhost:9999');
socket.on('news', function (data) {
          document.getElementById("chat").innerHTML +=
data.data + "</br>";
});
function send() {
 socket.emit('news', {data:document.getElementById("msg").
value});
}
</script>
</head>
```

The two parts to this are setting a callback for news events and the **send** function. When this page receives a news event, it just adds the content of the data portion of the JSON object to the chat ** **, and appends a line break. The **send** function creates a news event on the server, and attaches a JSON object that includes a data element which contains the content of the **msg** text input.

That's all you need to create a simple chat server. If you make these changes, and restart **nodejs app.js**, you'll be able to chat between two people using the web page. To test this out, open two web browsers (such as *Firefox* and *Chrome*), and connect both of them to **http://localhost:9999**, and you should be able to send messages between them.

Securing the data

What we've just created is probably the least secure chat tool ever. Not only can the server see every message that's being sent, so too can anyone else on the network and anyone else who happens to connect to the server. What's more, there's no authentication, so you've no idea who's sending messages.

We said at the start that we would implement end-to-end encryption to guarantee user's privacy, and that means that we have to encrypt the messages in the web browser, and not decrypt them again until they reach to the destination browser. Since the server just passes data around, it doesn't matter if this data is plaintext or ciphertext, so we can add this client-side encryption without changing the server at all.

For the encryption, we'll use *CryptoJS*. This is a library that implements a number of standard encryption techniques. The project is hosted on Google Code (**https://code.google.com/p/crypto-js**) however, Google Code is shutting down, and there's

Express.JS Getting to grips with a Node.js web framework.

We've barely touched on the power of the Express.js framework, which can do far more than just serve up HTML pages. It's based on the concept of middleware. In Express.js, middleware is a series of functions that run one after the other that can all access the request and response objects. One could, for example, log the request, while another could make a decision based on the device that sent the request. At the end of the middleware stack, the response object should be fully formed and can be sent. There's more information, and a guide to help get you started, on the express.js website: **http://expressjs.com**.

a mirror on GitHub (**https://github.com/sytelus/CryptoJS**).

First we need to add a text box for the user to enter the encryption key into the **<body>** section of the HTML. You can add this directly under the **<h1>** line:

```
<p>Encryption Key: <input id="key" type="text"></input>
```

Then you need to include the JavaScript library by adding the following script line just below the **<title>** line in **index.html**:

```
<script src="http://crypto-js.googlecode.com/svn/tags/3.1.2/build/rollups/aes.js"></script>
```

Then you can change the contents of the main **<script>** tag to;

```
<script>
var socket = io.connect('http://localhost:9999');
socket.on('news', function (data) {
 console.log(data);
 var decrypted = CryptoJS.AES.decrypt(data.data, document.getElementById("key").value);
 if (decrypted.toString(CryptoJS.enc.Utf8) != "") {
        document.getElementById("chat").innerHTML += decrypted.toString(CryptoJS.enc.Utf8) + "</br>";
 }
});
function send() {
 var encrypted = CryptoJS.AES.encrypt(document.getElementById("msg").value, document.getElementById("key").value);
 socket.emit('news', {data:encrypted.toString()});
}
</script>
```

As you can see, this works in exactly the same way as the unencrypted chat, but it uses CryptoJS's **encrypt** and **decrypt** functions to protect the messages using AES encryption before sending them. This is a highly secure encryption standard and should keep your messages safe from prying eyes.

Note that we've made no attempt to organise key exchange – this has to happen offline. Before two people can start chatting, they have to first agree on a key, then both go to this website at the same time (it doesn't store messages, so if two users aren't online at the same time, the message is lost).

What about security?

The server will send all messages to all the browsers connected; however, there's no guarantee that all browsers are using the same key to communicate. This is deliberate, and enables several groups of people to communicate anonymously on the same board. You will only be able to see the messages of people using the same key as you, but since all messages are sent to everyone, anyone monitoring the board can't prove who is chatting with whom. In fact, a spy can't even prove that you're chatting with anyone (even if there's data leaving your machine, they don't know if it's ever being decrypted).

There's a problem with this board. It will insert whatever a user types in the text box directly into the HTML of every other connecting client that's using the same encryption key. In normal usage, this would be text, but a malicious user could enter some HTML code, or even JavaScript, that attacks the other users.

For example, if someone enters:

```
<b onmouseover=alert('helloworld')>click me!</b>
```

They can inject a JavaScript alert (or potentially any other code) into the other client's browsers. This is known as a Cross Site Scripting (XSS) attack. In order to prevent them, we need to encode the contents of the textbox so that it shows as text, rather than HTML. To do this we need an encoder. We'll use the one at **www.strictly-software.com/scripts/downloads/encoder.js** rather than creating one ourselves. To use this, you need to change the **socket.on** callback in **index.html** to:

```
socket.on('news', function (data) {
 console.log(data);
 var decrypted = CryptoJS.AES.decrypt(data.data, document.getElementById("key").value);
 if (decrypted.toString(CryptoJS.enc.Utf8) != "") {
        var safeString = Encoder.htmlEncode(decrypted.toString(CryptoJS.enc.Utf8));
        document.getElementById("chat").innerHTML += safeString + "</br>";
 }
});
```

If you do this, whatever the users enter (special characters and all) will appear in the window, and any code won't be executed. If you make these changes, and restart your app, you should find that the XSS attack no longer works.

We set out to create a private and anonymous chat system. Our simple app hasn't been fully vetted by security experts, but we think that it fulfils this role well. It's private because it has end-to-end encryption that's supplied by (what we believe to be) a secure cryptographic library. It's anonymous because if you connect through *Tor*, no-one can tell if you're connected to it, and anyone monitoring the server (even the server operators themselves) can't tell who is speaking with whom. They only see encrypted text go back and forth, and have no way on knowing who's decrypting which messages. Because it's limited to text, it should scale well, but it won't scale indefinitely. It requires each user to have enough bandwidth to receive every message, and the server to be able to send every message to every connected person.

"Our simple app hasn't been fully vetted by security, but we think it fulfils its role well."

We should point out at this point that we're relying on CryptoJS and the Encoder for security, and we haven't fully vetted them for high security. If you want to test out this app, or make any changes, you can find the full code for the finished web app on GitHub at **https://github.com/linux-voice/issue17-node**.

Ben Everard isn't paranoid – they really are all out to get him.

LINUX VOICE

MASTERCLASS

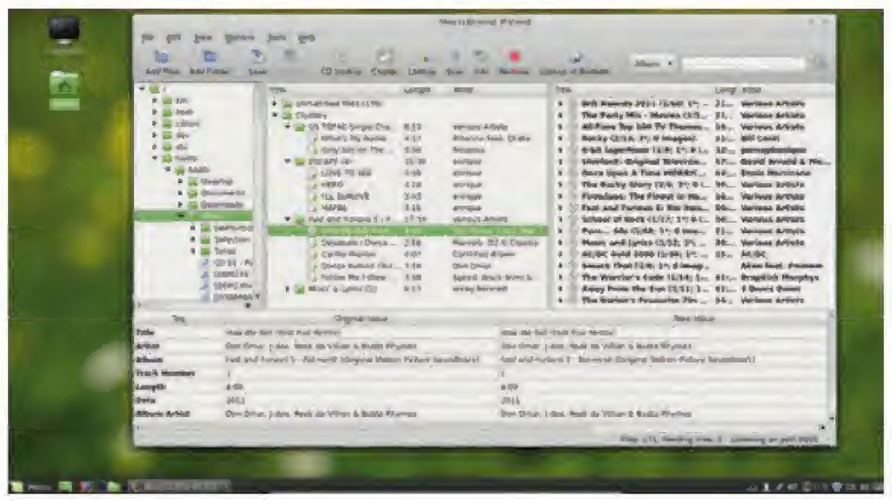
Undo the wrongs of your 90s self by tagging your anarchic music collection correctly.

TAKE CHARGE OF THE UNRULY LIBRARY WITH PICARD

If your music's grown like the Borg, you need a Picard to control it.

MAYANK SHARMA

Who doesn't love the sound of DRM-free music, or the gigabytes of music you've legally ripped from CDs and perhaps even LPs for your own listening pleasure? However, the one side-effect of owning so much music is disorganisation. *Picard* is a nifty little app that can help you get your music files back in shape. The app is designed to sort your music library and fill in missing tags, rename oddly named files and easily identify incomplete albums. *Picard* is developed by the MusicBrainz project, which is an online database that captures and hosts various information about artists and their recordings including track titles, album titles, length of each track, cover art, and more. According to statistics published by the project, their database contains information about 950,000 artists, 1.5 million releases, and 14.8 million recordings.



The idea with *Picard* is to get all your tracks into the right-most pane.

LV PRO TIP

If you have dozens of albums you may want to break the process down into several runs, at least until you're comfortable with *Picard*.

Picard to the bridge

The software has a simple layout, though you'll have to use it a couple of time to inculcate its methodology. Start by adding a handful of music files. *Picard* supports all popular formats including Ogg, Speex, Opus, Flac, MP3, WMA and WAV. You can drag the files from the file manager into the left-hand pane of *Picard*'s main interface, or head to View > File Browser to search your filesystem from within *Picard* itself.

"Picard is a nifty little app that can help you get your music files back in shape."

Any new files you add will initially be added under the Unmatched Files folder. Click on one of these files to view its current metadata in the bottom panel. You can right-click on any of the fields and select Edit to manually modify the tags.

The first step to get your music organised is to cluster all files from the same album under one heading. For this, select the Unmatched Files folder or any of the files you've just added and click the Cluster button on the toolbar. Depending on the album metadata present for each of the files, *Picard* will group all songs from the same album together. Files that don't belong in a clustered album will be listed as Unmatched Files.

Review the clusters, and if a file has been wrongly added, you can drag it to another cluster or back into the Unmatched Files folder. Similarly, you can manually drag any unmatched files into the correct cluster if you know they belong there.

Once you've clustered the files based on the existing metadata, you can either click Lookup or Scan for matching the files with the online database. The difference between the two options is that a lookup is done on an entire cluster at once, and uses whatever existing metadata is already in the files to query the database. It's quite fast and usually gets the job done. If *Picard*'s lookup doesn't work, you'll have to scan the files. Scanning is done on a file-by-file process, and uses each file's audio fingerprint. It takes much longer, but it works well.

After fetching the relevant information, *Picard* displays the name of the album that every song

belongs to in the panel on the right. The app makes intelligent guesses to pair the track with an album. It also removes these songs from the Unmatched Files list and moves them into this new album entry in the panel on the right. Expand each album to view your track (or tracks), which will be marked with a small colour-coded rectangular icon. Green is a good match; yellow, orange and red represent increasing degrees of uncertainty. It also uses a colour-coded scheme to point to album completeness. Complete albums are shown with a golden icon. The ones with a silver icon are missing some tracks, which is clearly noted next to the album's name.

Inspect each album and track. If the software has identified it correctly, click on the Save button. This tells *Picard* to attach the new metadata to this song. As a visual indicator, the coloured rectangle changes to a green checkmark to show the track information has been saved.

Disengage autopilot

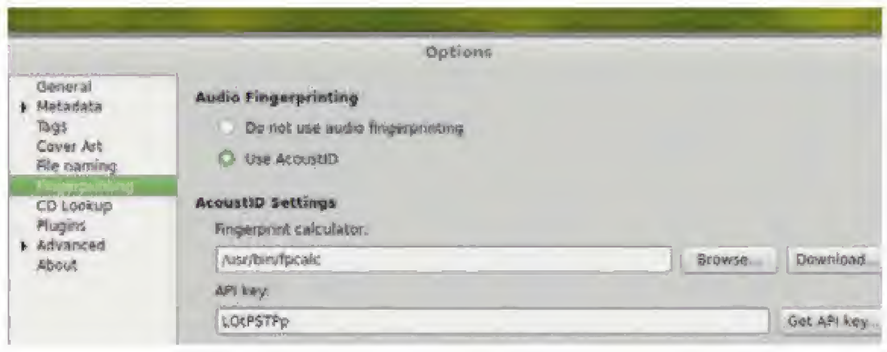
For songs that are matched incorrectly, you may find they've been confused with another track in the same album. In this case, you can manually rearrange the tracks into the correct order by dragging and dropping them. If the data is completely wrong, however, select the song and drag it back to the Unmatched Files folder. To improve the chances of detection, add some information to the song yourself (the artist's name, track length or album name are helpful). After that, click on the Lookup icon in the toolbar at the top again and *Picard* is more likely to find an appropriate match.

If the program can't find the data automatically, you can also find track information manually by right-clicking the file and selecting the Lookup in Browser option. This will fire up your browser and point you in the direction of the MusicBrainz database, which shows you a list of all possible matches. You can also query this database by visiting the MusicBrainz website at **musicbrainz.org**. When you find an entry matching your track, click the Tag button adjacent to the album name and a new folder should pop up in the right-hand side of the *Picard* window.

When you're done, hit the Save button in the toolbar to associate the new metadata with the files.

You don't need to tweak *Picard* in any way to fill in the tags for your music collection. However, once you

You can extend *Picard*'s already impressive functionality by enabling a variety of plugins.



Head to the AcoustID website and get an API key to contribute signatures of new tracks to its database.

get the hang of its basic operations, you might want to tweak some its options for a better experience.

Head to Option > Options to access all of *Picard*'s settings under various tabs. The Automatically Scan All New Files option under the General tab will save you a click by automatically looking up tracks as soon as you add them in *Picard*. Then there's the Metadata tab, which has an option to translate foreign artists' names into English. If you shuttle your tracks between devices, switch to the Tags tab and make sure you set the Tag Compatibility to 2.3, which has a broader support that the newer v2.4.

The most interesting option is under the File Naming tab. Although *Picard* uses the metadata to update each file's tags, you can also have it rename the files and place them in folders according to the naming scheme you see fit. This functionality is disabled by default. You can enable it by toggling the Rename Files When Saving option under the File Naming tab.

Furthermore you can also customise how *Picard* formats the file name by specifying a pattern in *Picard*'s own scripting language.If you use the **%artist% - %title%** pattern, *Picard* will format the name as The Beatles – Ticket to Ride.mp3. When you specify a pattern, *Picard* will preview how it will name the files using sample tracks in the Examples section below the pattern window.

Here's an interesting naming pattern adapted from an example we found on the web:

```
$replace($if($eq($left(%albumartist%,4),The),%albumartistsort%,%albumartist%)/%albumartist%-$left(%date%,4)-%album%$if(%discnumber%,-CD %discnumber%,)/$num(%tracknumber%,2)-%title%,:,)
```

This string will first generate a folder per artist using the Album Artist tag. If the tag begins with the word "The" it'll strip that string and stick it at the end. This means "The Rolling Stones" will be filed under "Rolling Stones, The" which makes it easy to sort the library in the regular folder browser.

Underneath the artist's folder we ask *Picard* to create one folder for each album. This folder will be named in the format <Album Artist>-<Year>-<Album Title>. If the album has multiple discs, the disc number will be inserted into the folder name, such as The Rolling Stones-1971-Sticky Fingers-CD 1.

When you hit the Save button after enabling the file renaming option, *Picard* will save the tags and also rename your tracks as per the naming pattern. You can find various examples of *Picard*'s renaming patterns on its wiki; it does take some getting used to. but will help you better sort and organise your music.

LV PRO TIP

Drag a directory from the file browser to an album – this attempts to match all the files from the directory to the album.

LV PRO TIP

Here's a list of *Picard*'s internal tag name and their equivalents in other tagging formats (**http://picard.musicbrainz.org/docs/mappings**).

FIX THE TUNES FROM THE CLI

Beat your library into shape with Beets.

MAYANK SHARMA

Picard is a fabulous application. But its biggest shortcoming is that it's a graphical app! If you want to fix the incomplete tags in your music library from the confines of the familiar, versatile and venerable command line interface, you need *Beets*, which bills itself as the music geek's media organiser.

Beets is available in the repositories of most popular distributions. However, the version in your distro's repository might not be the latest. The recommended way to install *Beets* is via *PIP*, which is a package management system used exclusively for installing packages written in Python. On Deb-based systems such as Ubuntu, install *PIP* and its dependencies with

```
sudo apt-get install python-pip
```

and on RPM-based distros with

```
sudo yum install python-pip
```

Once installed, you can use *PIP* to install *Beets* with

```
sudo pip install beets
```

LV PRO TIP

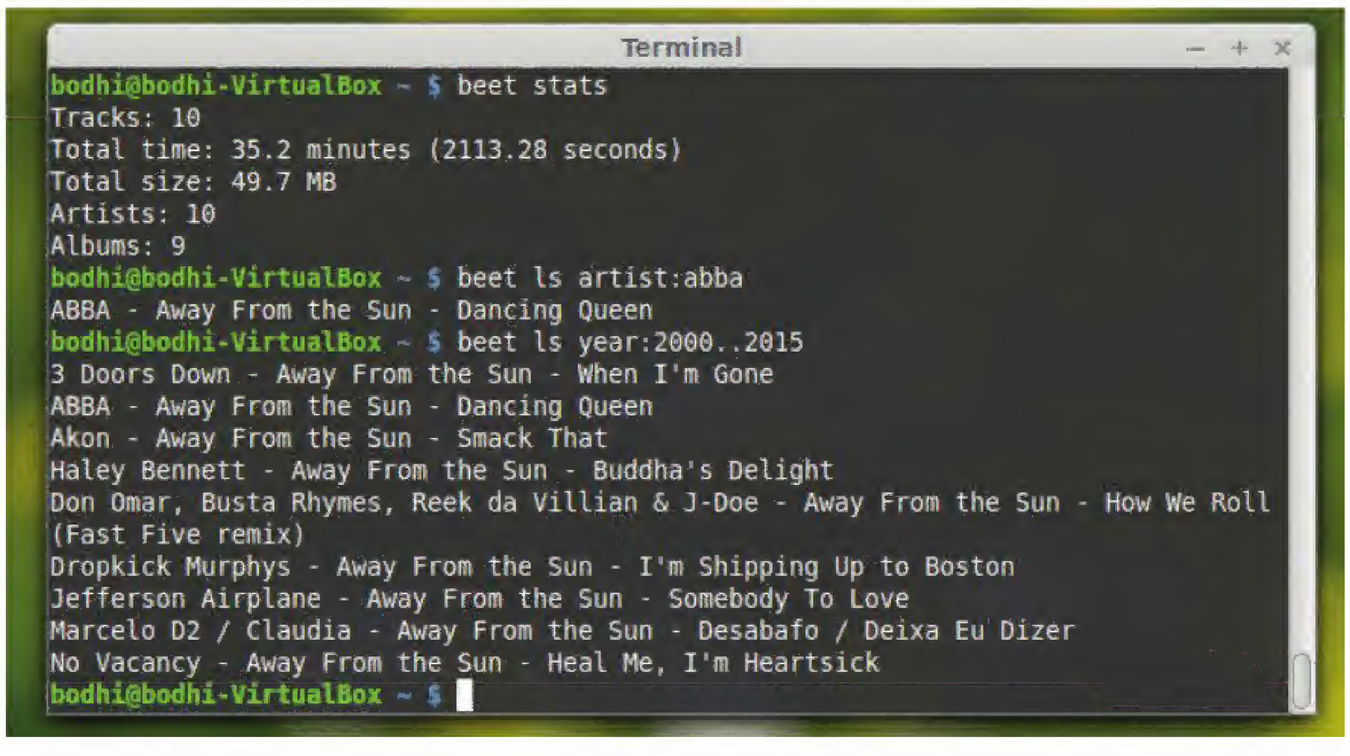
The tool is called *Beets* but you interact with it using the **beet** command.

Configure Beets

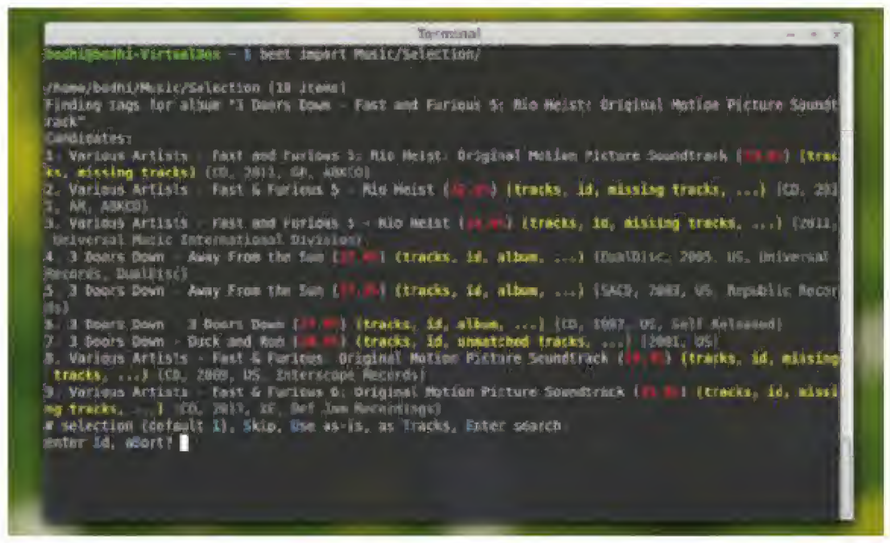
Unlike most command line tools that ship with a working configuration file, you have to manually create the configuration file for *Beets*. The configuration is stored in a text file called **config.yaml** that's placed under the **~/.config/beets/** directory. The configuration file will grow with time as you become more familiar with *Beets*. To begin with, start with the following entries:

```
directory: ~/Music/beets-music
library: ~/Music/beets-music/musiclibrary.blb
```

Make sure you create the **~/Music/beets-music/** directory beforehand. The **directory** option points to the directory where you wish to store your music collection. Remember, this isn't the path to your existing music collection. Rather, this is the directory where *Beets* will store your music after the tool has imported and reorganised it. The **library** path is where *Beets* will store the database file that stores the index metadata of your music files.



Use the **beet fields** command for a complete list of items and album fields that you can use in your queries.



Beets resolves conflicts between settings in the config file and the CLI by going with the latter.

After creating the config file, we need to import our music collection into *Beets*. By default, the tool assumes that we'll organise all our music under the directory specified in the configuration file. You can either copy the music from where it resides currently, or you can move the music to save disk space.

If you wish to move the files into the specified directory, add these lines to the configuration file:

```
import:
    move: yes
```

On the other hand, if you don't wish to change the location of your music files, you can specify the path to your music files in the **directory** option and use the following lines to inform *Beets* that you don't wish to copy or move files from the current location:

```
import:
    copy: no
    move: no
```

Make note of the indents. The configuration file is in the YAML language, which accepts spaces (and not tabs) to indent some lines. Refer to the official documentation for more configuration options (**http://beets.readthedocs.org/en/v1.3.13/reference/config.html**)

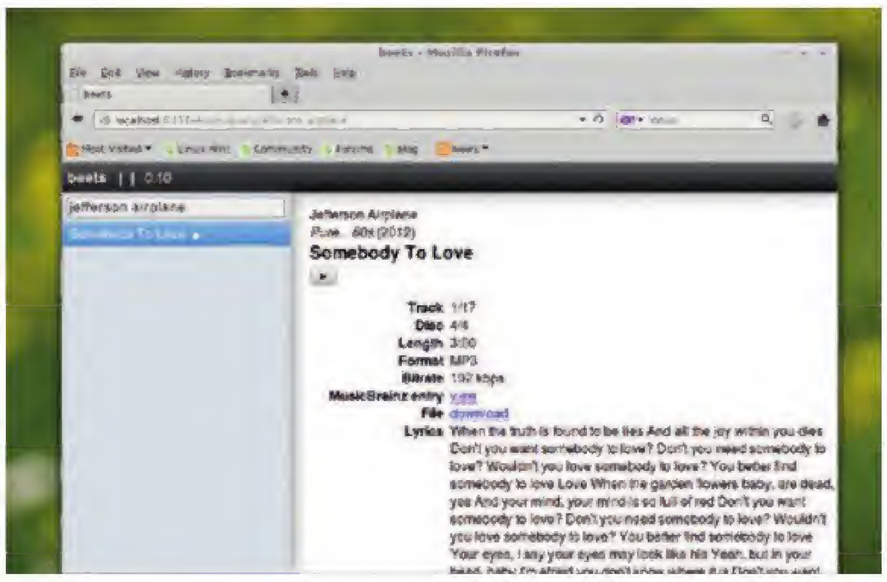
Import music

With the configuration file in place, you're finally ready to import files. The command

```
beet import /path/to/music/files
```

will import your music. The command also copies or moves the files into your specified directory depending on the import options you've specified in the configuration file. As the **import** command brings in your music, it'll also fix and fill any gaps in the metadata from the MusicBrainz database.

If it finds multiple choices for a track, *Beets* will let you select the one that matches the track. If none of the candidates match your album, press U, which tells *Beets* to import files as it is. Note that the import process does not produce any output on the screen,



The web interface is pretty basic but gets the job done.

except for when it lists the possible candidates based on metadata. The process can take a long time, so if you wish to import a large selection of files in one go, and don't want to be prompted again and again for the metadata information, use the **-A** option, such as

```
beet import -A /path/to/many/music/files/
```

which asks *Beets* not to auto-tag the files and is much faster.

If you haven't modified the configuration file to instruct *Beets* on how to handle the imported music, it'll stick to its default behaviour and copy the music. Use the **-C** option when importing music to ask *Beets* to update the tags without copying the music.

Browse the library

After importing the files, you can use the **beet ls** command to query the music library. This command expects a query string, and if you don't specify any delimiters, it will search all the metadata files for the supplied query string. So for example, if you use the **beet ls Rocky II** command, *Beets* will list all songs where both the words Rocky and II appear in the metadata, whether in the title, album, artist, and so on.

When you supply multiple keywords as the query string, the words are automatically joined with a Boolean AND operator. That is, *Beets* will only display results where both the words appear.

You can also restrict the searches to specific fields such as artist, album and year, such as **beet ls artist:Beatles** or **beet ls year:2004**. You can also specify a numeric range as a query, and combine it with other list options. For example,

```
beet ls -a year:2004..2005
```

will list all albums released between 2004 and 2005. The **-a** option queries albums instead of individual tracks. The command

```
beet ls format:MP3 bitrate:128000
```

will list all MP3s where the bitrate is more than 128k. Refer to the official online documentation (**https://beets.readthedocs.org/en/v1.3.13/reference/query.html**) for more query options.

Extend Beets

You can extend the core functionality of *Beets* with plugins. *Beets* ships with several plugins by default, but they need to be enabled before you can use them.

If you'd like to fetch lyrics for songs, retrieve cover art for albums, and provide new metadata sources, and more, while *Beets* imports your music, you can enable the concerned plugins by editing the config file.

```
$ nano ~/.config/beets/config.yaml
plugins: lyrics fetchart scrub
```

The **plugins:** line expects a space-separated list of all the plugins you wish to enable. In the above example, the Lyrics, FetchArt and the Scrub plugin are enabled. You can also use the

```
beet lyrics <song name>
```

command to manually search for lyrics for a song. *Beets* will automatically store the lyrics in the database. You can then use the

```
beet lyrics -p <song name>
```

command to print the lyrics on the screen. *Beets* will first search for the lyrics in the database, and if it doesn't find a match, it'll fetch the lyrics from the online sources.

Beets also has a simple web UI. To use the web interface you need the Flask framework, which you can install with

```
sudo pip install flask
```

Then put **web** in the plugins line in the configuration file and start the web server with

```
beet web
```

Now launch your web browser and head to **http://localhost:8337** to access the interface. Using the web interface you can search through your imported music collection. Click on a song from the results to view its metadata including the lyrics if you've enabled the plugin and fetched them. The web interface also has basic controls to play and pause music.

Beets can also fingerprint your music and query the AcoustID database to find a match. First grab the *Chromaprint* library (**https://acoustid.org/chromaprint**) for your computer's architecture and extract it to reveal the *fpcalc* binary, which you should place in **/usr/local/bin/** with

```
sudo mv ~/Downloads/fpcalc /usr/local/bin/
```

Then install the dependencies for the plugin with

```
sudo apt-get install python-gst0.10-dev
```

and then install the plugin using *PIP* with

```
sudo pip install pyacoustid
```

Once you have installed the dependencies, enable the plugin by adding the word **chroma** and rerun the **import** command to generate and match the signatures for the music in your library.

Both *Beets* and *Picard* are feature-rich tools that you can use to organise your dishevelled music library. Even though we've covered some of their most useful and interesting plugins, they offer a lot more options than what's mentioned here. You can use either tool based on the level of comfort of their respective operational environments, though *Beets* can do everything *Picard* can and more.

Mayank Sharma has been finding productive new ways to mess about with free software for years now.

LV PRO TIP

Use **beet ? import** to list all the import options that you can use.

LV PRO TIP

Use **beet stats** to get statistics about your collection including the total number of tracks, total number of artists, the total play time and more.

/DEV/RANDOM/

Final thoughts, musings and reflections

Nick Veitch was the original editor of Linux Format, a role he played until he got bored and went to work at Canonical instead. Splitter!

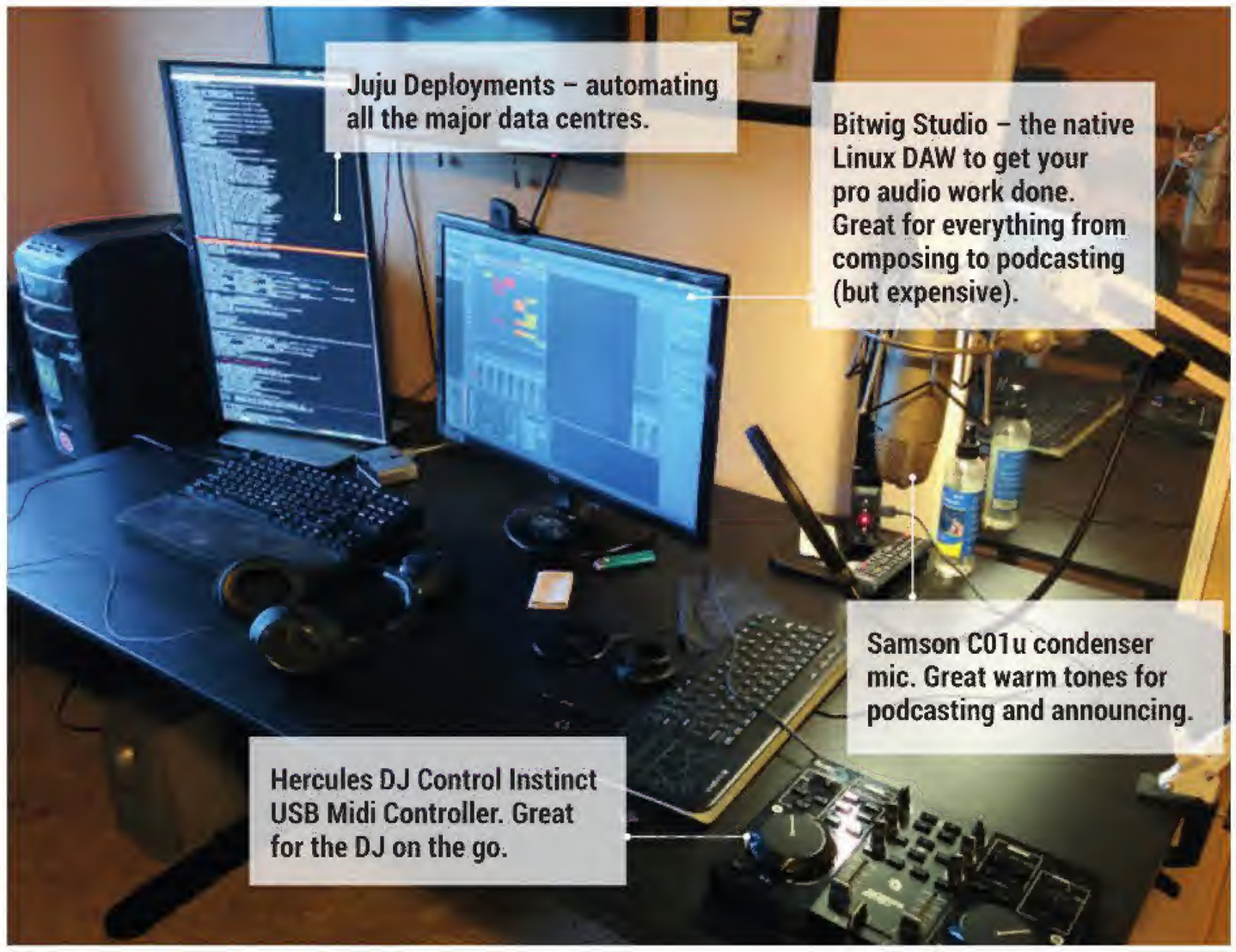
Recently on Twitter I congratulated the Linux Voice readership on their prescient choice to donate some of the LV profit hoard towards the Open Rights Group (**www.openrightsgroup.org**), an organisation dedicated to protecting the digital rights of the individual.

It was prescient, because the UK has elected a new government, one which, unfettered by any sort of liberal conscience, wishes to pursue a new "Investigatory Powers Bill", or the "Snooper's Charter" as some have dubbed it.

It remains to be seen what specific Orwellian fantasy will manifest itself through such a bill, but based on previous efforts, it seems likely it will embody the right of the state to basically read and spy on absolutely everything you do online, with no accountability or justification. Mandatory state-controlled backdoors in messaging systems may seem like a grand idea to combat terrorists, but *quis*, as ever, *custodiet*?

We can't even rely on the age-old defence against government meddling – their own incompetence. When David Cameron announced he was going to clean up the internet by blocking things he considered undesirable, the result was the predicted catastrophe. But where a set of skeleton keys for encryption tech is concerned, the foreknowledge that it will be bungled isn't much compensation – if I were to bet on who would get my bank details first, I'm pretty sure I'd rate the hacker skills of global terrorists over those of the cabinet office.

So, if you didn't vote to donate to ORG, there's still time to take a look at the website linked above and get involved with some very sensible campaigns. But gosh, only if you believe in freedom, openness and transparency of course. And if you are a terrorist, I would like to point out that David Cameron has more money than I do.



My Linux Setup Charles Butler

Professional tech dabbler, Juju Charms wrangler and more.

Q What version of Linux are you currently using?

A Ubuntu 14.04 – I love how ubiquitous Ubuntu is, on all hardware, everywhere. Most everything 'just works' these days with it, and I value that above the enablement story, above anything else in terms of eye candy.

Q And what desktop do you currently use?

A Unity. I'm a very keyboard-focused user. Every day I'm flying in and out of contexts between servers, my desktop apps, and appliances. Having a keyboard-centric environment is crucial, especially when doing live streaming shows. Reaching for the mouse can mean I miss a critical transition.

Q What was the first Linux setup you ever used?

A In the mid to late 90s I received a SAMS Linux admin guide with three distros. It's hard to remember whether it was Red Hat 5.1, Slackware 6, or Corel Linux – but one of those three.

Q What Free Software/open source can't you live without?

A *Internet DJ Console* and *Icecast* - between these two tech's I've managed to reach over 400 thousand listeners. As an indie DJ, these tools are reliable and pro grade.

Q What do other people love but you can't get on with?

A I'm going to have to go there... but *Emacs*. I'm a *Vim* user, through and through – but I don't openly hate on *Emacs*, to each their own.

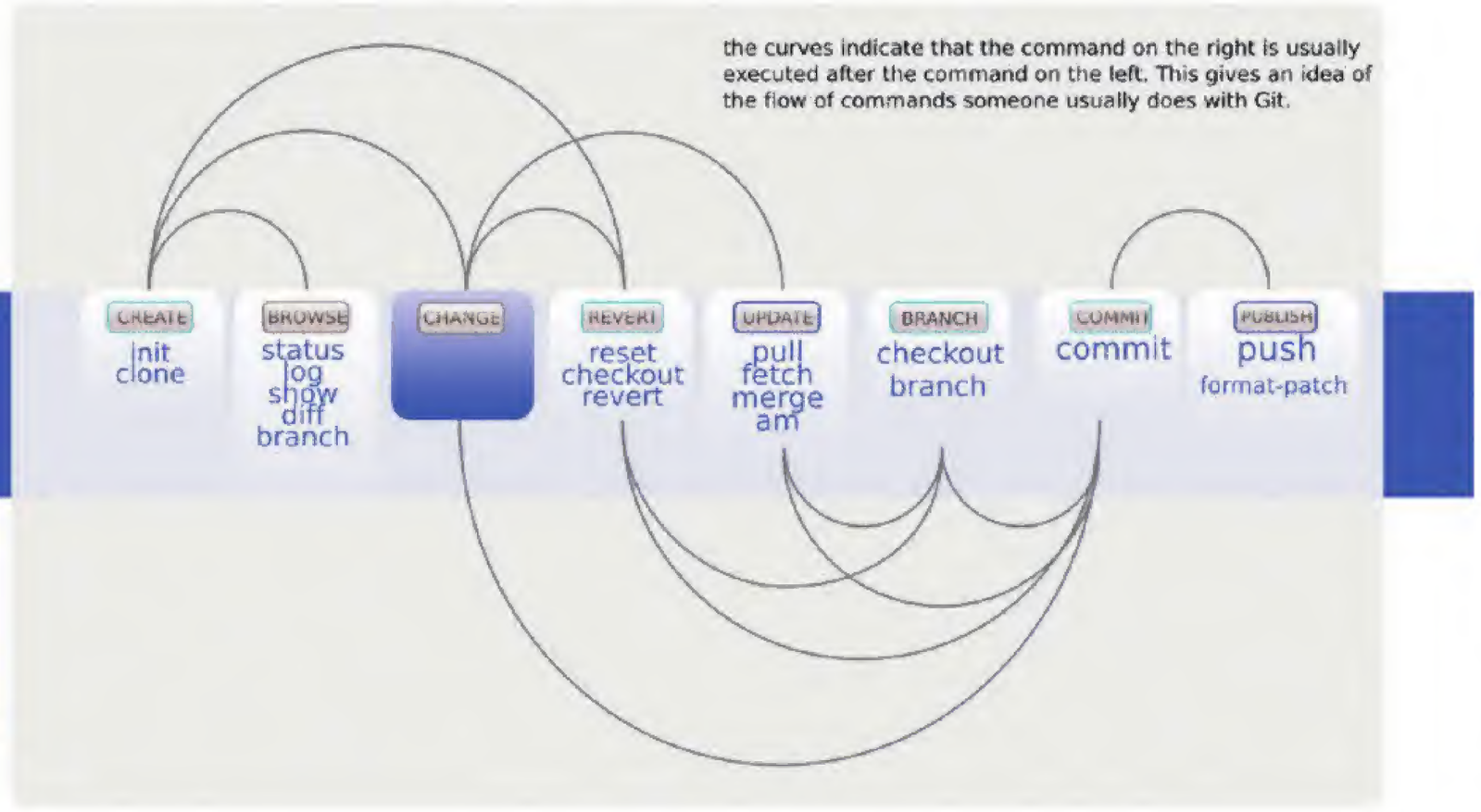
Git Cheat Sheet

http://git.or.cz/

Remember: git command --help

Global Git configuration is stored in $HOME/.gitconfig (git config --help)

Commands Sequence



Create

From existing data

```
cd ~/projects/myproject
git init
git add .
```

From existing repo

```
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone ssh://you@host.org/proj.git
```

Show

Files changed in working directory

```
git status
```

Changes to tracked files

```
git diff
```

What changed between $ID1 and $ID2

```
git diff $id1  $id2
```

History of changes

```
git log
```

History of changes for file with diffs

```
git log -p $file $dir/ec/tory/
```

Who changed what and when in a file

```
git blame $file
```

A commit identified by $ID

```
git show $id
```

A specific file from a specific $ID

```
git show $id:$file
```

All local branches

```
git branch
```

(star '*' marks the current branch)

Cheat Sheet Notation

$id : notation used in this sheet to represent either a commit id, branch or a tag name
$file : arbitrary file name
$branch : arbitrary branch name

Concepts

Git Basics

```
master    : default development branch
origin    : default upstream repository
HEAD      : current branch
HEAD^     : parent of HEAD
HEAD~4    : the great-great grandparent of HEAD
```

Revert

Return to the last committed state

```
git reset --hard
```

⚠ you cannot undo a hard reset

Revert the last commit

```
git revert HEAD
```

Creates a new commit

Revert specific commit

```
git revert $id
```

Creates a new commit

Fix the last commit

```
git commit -a --amend
```

(after editing the broken files)

Checkout the $id version of a file

```
git checkout $id $file
```

Branch

Switch to the $id branch

```
git checkout $id
```

Merge branch1 into branch2

```
git checkout $branch2
git merge branch1
```

Create branch named $branch based on the HEAD

```
git branch $branch
```

Create branch $new_branch based on branch $other and switch to it

```
git checkout -b $new_branch $other
```

Delete branch $branch

```
git branch -d $branch
```

Update

Fetch latest changes from origin

```
git fetch
```

(but this does not merge them)

Pull latest changes from origin

```
git pull
```

(does a fetch followed by a merge)

Apply a patch that some sent you

```
git am -3 patch.mbox
```

(in case of a conflict, resolve and use `git am --resolved`)

Publish

Commit all your local changes

```
git commit -a
```

Prepare a patch for other developers

```
git format-patch origin
```

Push changes to origin

```
git push
```

Mark a version / milestone

```
git tag v1.0
```

Useful Commands

Finding regressions

```
git bisect start      (to start)
git bisect good $id   ($id is the last working version)
git bisect bad $id    ($id is a broken version)

git bisect bad/good   (to mark it as bad or good)
git bisect visualize  (to launch gitk and mark it)
git bisect reset      (once you're done)
```

Check for errors and cleanup repository

```
git fsck
git gc --prune
```

Search working directory for foo()

```
git grep "foo()"
```

Resolve Merge Conflicts

To view the merge conclicts

```
git diff          (complete conflict diff)
git diff --base  $file  (against base file)
git diff --ours  $file  (against your changes)
git diff --theirs $file  (against other changes)
```

To discard conflicting patch

```
git reset --hard
git rebase --skip
```

After resolving conflicts, merge with

```
git add $conflicting_file  (do for all resolved files)
git rebase --continue
```

Zack Rusin
Based on the work of
Sébastien Pierre
Xiaoma Corp